

Emergence, Complexity and Computation ECC

Pierluigi Frisco  
Marian Gheorghe  
Mario J. Pérez-Jiménez *Editors*

# Applications of Membrane Computing in Systems and Synthetic Biology

 Springer

# Emergence, Complexity and Computation

## Volume 7

### *Series editors*

Ivan Zelinka, Technical University of Ostrava, Ostrava, Czech Republic  
e-mail: [ivan.zelinka@vsb.cz](mailto:ivan.zelinka@vsb.cz)

Andrew Adamatzky, University of the West of England, Bristol, UK  
e-mail: [adamatzky@gmail.com](mailto:adamatzky@gmail.com)

Guanrong Chen, City University of Hong Kong, Hong Kong  
e-mail: [eegchen@cityu.edu.hk](mailto:eegchen@cityu.edu.hk)

### *Editorial Board*

Ajith Abraham, MirLabs, USA

Ana Lucia C. Bazzan, Universidade Federal do Rio Grande do Sul, Porto Alegre  
RS Brasil

Juan C. Burguillo, University of Vigo, Spain

Sergej Čelikovský, Academy of Sciences of the Czech Republic, Czech Republic

Mohammed Chadli, University of Jules Verne, France

Emilio Corchado, University of Salamanca, Spain

Donald Davendra, Technical University of Ostrava, Czech Republic

Andrew Ilachinski, Center for Naval Analyses, USA

Jouni Lampinen, University of Vaasa, Finland

Martin Middendorf, University of Leipzig, Germany

Edward Ott, University of Maryland, USA

Linqiang Pan, Huazhong University of Science and Technology, Wuhan, China

Gheorghe Păun, Romanian Academy, Bucharest, Romania

Hendrik Richter, HTWK Leipzig University of Applied Sciences, Germany

Juan A. Rodriguez-Aguilar, IIIA-CSIC, Spain

Otto Rössler, Institute of Physical and Theoretical Chemistry, Tübingen, Germany

Vaclav Snasel, Technical University of Ostrava, Czech Republic

Ivo Vondrák, Technical University of Ostrava, Czech Republic

Hector Zenil, Karolinska Institute, Sweden

For further volumes:

<http://www.springer.com/series/10624>

### *About this Series*

The Emergence, Complexity and Computation (ECC) series publishes new developments, advancements and selected topics in the fields of complexity, computation and emergence. The series focuses on all aspects of reality-based computation approaches from an interdisciplinary point of view especially from applied sciences, biology, physics, or Chemistry. It presents new ideas and interdisciplinary insight on the mutual intersection of subareas of computation, complexity and emergence and its impact and limits to any computing based on physical limits (thermodynamic and quantum limits, Bremermann's limit, Seth Lloyd limits...) as well as algorithmic limits (Gödel's proof and its impact on calculation, algorithmic complexity, the Chaitin's Omega number and Kolmogorov complexity, non-traditional calculations like Turing machine process and its consequences,...) and limitations arising in artificial intelligence field. The topics are (but not limited to) membrane computing, DNA computing, immune computing, quantum computing, swarm computing, analogic computing, chaos computing and computing on the edge of chaos, computational aspects of dynamics of complex systems (systems with self-organization, multiagent systems, cellular automata, artificial life,...), emergence of complex systems and its computational aspects, and agent based computation. The main aim of this series is to discuss the above mentioned topics from an interdisciplinary point of view and present new ideas coming from mutual intersection of classical as well as modern methods of computation. Within the scope of the series are monographs, lecture notes, selected contributions from specialized conferences and workshops, special contribution from international experts.

Pierluigi Frisco · Marian Gheorghe  
Mario J. Pérez-Jiménez  
Editors

# Applications of Membrane Computing in Systems and Synthetic Biology



*Editors*

Pierluigi Frisco  
School of Mathematical and Computer  
Sciences  
Heriot-Watt University  
Edinburgh  
UK

Mario J. Pérez-Jiménez  
Department of Computer Science and  
Artificial Intelligence  
University of Sevilla  
Sevilla  
Spain

Marian Gheorghe  
The Department of Computer Science  
University of Sheffield  
Sheffield  
UK

ISSN 2194-7287

ISBN 978-3-319-03190-3

DOI 10.1007/978-3-319-03191-0

Springer Cham Heidelberg New York Dordrecht London

ISSN 2194-7295 (electronic)

ISBN 978-3-319-03191-0 (eBook)

Library of Congress Control Number: 2013955251

© Springer International Publishing Switzerland 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

*To Gheorghe Păun*

# Preface

Membrane Computing was introduced in 2000 [1], as a research topic in Natural Computing. It soon became a very active research area and just three years after the publication of this seminal paper, the Institute for Scientific Information (ISI) identified the paper as being *fast breaking* in the field of Computer Science [2]. Since then the field has flourished giving birth to many research topics, triggering connections with many other research areas and producing interesting applications. The models introduced are called membrane (or P) systems. Many theoretical aspects related to computability, complexity and decidability have been investigated for various classes of membrane systems—accounts of these developments have been reported in [3–6]. Membrane systems have been employed for solving various problems in Computer Science, Graphics, Linguistics, Robotics, and other fields [7]. A recently published comprehensive handbook [8] provides a thorough overview of the main theoretical developments, it points to the main links between membrane systems and other computational models and describes some of the most important applications in this field and some of the tools developed throughout the years.

In parallel to investigations of the key research questions related to Membrane Computing and their applications, there have been studies related to the use of the Membrane Computing paradigm in modelling biological systems. These applications represent not only significant contributions to modelling various processes and phenomena in Biology, but they also point out interesting and challenging problems in studying the complexity and emergent properties of such systems. Some of the contributions related to these applications have been collected in a special issue of BioSystems [9], but there is no monograph dedicated to such issues, and this is a clear gap in the literature of membrane systems. The present monograph aims to fill this gap.

The authors invited to write chapters for this volume have already published papers on a broad spectrum of problems related to Biology, from modelling and simulations, to system analysis and identification, and to the development of methods and tools necessary in supporting these investigations. There have been reported studies on modelling oscillating and catalytic chemical reactions [10], diffusing processes with geometrically constrained molecules [11] or occurring in a signal transduction pathway [12]. Stable oscillatory states in budding yeast [13], biochemical signalling pathways [14] and Fas pathways [15] have been considered.

Classes of probabilistic membrane systems have been introduced for studying populations and ecosystems [16] and metabolic P systems, a special class of deterministic systems, have been utilised in solving complex problems in biotic systems [17, 18]. Membrane systems have been used to model and simulate the behaviour of logical gates [19] and register machines [10] in a synthetic biology setting. Slightly more general models, like agent-based approaches, have been utilised for studying three-dimensional multiscale phenomena in human epidermis [20]. The definition of formal languages for specifying complex molecular interactions in biological systems [21], verification tools for non-deterministic [22] and stochastic systems [23], and the development of complex platforms for specification, analysis and simulation of systems and synthetic biology problems [24] have been considered. Appropriate sets of methods and tools [17, 18] have been developed and efficient implementations of classes of membrane systems for complex simulations [25] have been produced.

The chapters contained in this monograph give a clear image of the depth and breadth of the applications of membrane systems for the study of various biological processes and phenomena.

- In [Chap. 1](#), *Infobiotics Workbench: A P Systems-Based Tool for Systems and Synthetic Biology*, a comprehensive overview of an integrate software platform, the Infobiotics Workbench, and of its usage in specifying, simulation, verification and parameter optimisation of models operating at the cellular and intercellular levels is provided.
- In [Chap. 2](#), *Statistical Model Checking of Membrane Systems with Peripheral Proteins: Quantifying the Role of Estrogen in Cellular Mitosis and DNA Damage*, a methodology and a software platform for integrating a membrane system simulator with a statistical model checker are described. These are used for studying the dosage of antagonist that minimises the uncontrolled replication of abnormal cells.
- In [Chap. 3](#), *Molecular Diffusion and Compartmentalisation in Signal Transduction Pathways: An Application of Membrane Systems to the Study of Bacterial Chemotaxis*, intracellular diffusion processes are studied. A class of membrane systems, called  $\tau$ -DPP, is used in analysing both single volume pathways and multivolume diffusion interactions.
- In [Chap. 4](#), *Membrane Systems-Based Models for Specifying Dynamical Population Systems*, a class of probabilistic systems for studying population dynamics of ecosystems is presented. The theory behind the simulation of such models is presented, a software platform allowing the specification, simulation on different platforms, including CUDA, and analysis of such systems is described and a complex case study analysed.
- In [Chap. 5](#), *Membrane Systems and Tools Combining Dynamical Structures with Reaction Kinetics for Applications in Chronobiology*, three relevant studies in chronobiology are addressed, proving their convergent results. These case studies are specified and analysed with a software package, called SRSim, allowing spatial interaction rules and a powerful visualisation engine.

- In [Chap. 6](#), *Biochemical Networks Discrete Modelling Inspired by Membrane Systems*, a new version of Gillespie algorithm is presented. Comparisons between this model and others based on differential equations and Gillespie method are performed on a number of case studies.
- In [Chap. 7](#), *MP Modelling for Systems Biology: Two Case Studies*, a class of deterministic membrane systems, namely metabolic P (MP) systems, is described. The modelling capabilities of such models, expanding from metabolic systems to more general dynamical systems, and the power of a regression algorithm allowing the identification of MP models from the time series associated with observations are studied.
- In [Chap. 8](#), *Modelling and Analysis of E.coli Respiratory Chain*, a more general modelling approach, based on agent systems, and a method to derive a class of membrane systems specifications, called kernel P systems are presented. This process, which allows the formal verification of such specifications by using model checking methods, is illustrated by a prototype model of *E.coli* respiratory chain.

The entire description of all the examples discussed in this monograph together with simulation results and tools utilised are provided as auxiliary materials by the authors. Each chapter contains a link to webpages describing the case study(ies) presented.

The book is addressed to researchers interested in applications of discrete models in Biology, the interplay between membrane systems and other approaches and methods in specifying and analysing complex systems and revealing their behaviour. The readers are encouraged to use and assess the models described in the book chapters and the tools provided. Comments and suggestions for improving the functionality of the models and the usability of the tools are welcome.

This book is dedicated to Gheorghe Păun, the initiator and the main driving force behind the research in Membrane Computing. We thank Gheorghe for his continuous support, contagious enthusiasm and for being an example and source of inspiration.

We acknowledge the support and advice of our colleagues, Erzsébet Csuha-Jarj and Oscar Ibarra, during the process of elaborating this book and to thank the publisher for a friendly and efficient collaboration.

Pierluigi Frisco  
Marian Gheorghe  
Mario J. Pérez-Jiménez

## References

1. G. Păun, Computing with membranes. *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000)
2. Fast Breaking Paper, <http://esi-topics.com/fbp/fbp-february2003.html>.
3. G. Păun, *Membrane Computing. An Introduction* (Springer-Verlag, Berlin, 2002)
4. A. Păun, *Computability of the DNA and Cells. Splicing and Membrane Computing* (SBEB Publishing, Ruston, 2008)
5. P. Frisco, *Computing with Cells. Advances in Membrane Computing* (Oxford University Press, Oxford, 2009)
6. G. Ciobanu, *Membrane Computing. Biologically Inspired Process Calculi* (The Publishing House of the “A.I.I. Cuza” University, Iași, 2010)
7. G. Ciobanu, G. Păun, M.J. Pérez-Jiménez, *Applications of Membrane Computing* (Springer-Verlag, Berlin, 2006)
8. G. Păun, G. Rozenberg, A. Salomaa, *The Oxford Handbook of Membrane Computing* (Oxford University Press, Oxford, 2010)
9. M. Gheorghe, N. Krasnogor, M. Camara, P systems applications to systems biology. *BioSyst.* **91**(3) (2008)
10. T. Hinze, R. Fassler, T. Lenser, P. Dittrich, Register machine computations on binary numbers by oscillating and catalytic chemical reactions modelled using mass-action kinetics. *Int. J. Found. Comput. Sci.* **20**, 411–426 (2009)
11. G. Gruenert, B. Ibrahim, T. Lenser, M. Lohel, T. Hinze, P. Dittrich, Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinform.* **11**, 307 (2010)
12. D. Besozzi, P. Cazzaniga, S. Cocolo, G. Mauri, D. Pescini, Modeling diffusion in a signal transduction pathway: the use of virtual volumes in P systems. *Int. J. Found. Comput. Sci.* **22**, 89–96 (2011)
13. D. Pescini, P. Cazzaniga, D. Besozzi, G. Mauri, L. Amigoni, S. Colombo, E. Martegani, Simulation of the Ras/cAMP/PKA pathway in budding yeast highlights the establishment of stable oscillatory states. *Biotechnol. Adv.* **30**, 99–107 (2012)
14. J. Jack, A. Păun, Discrete modelling of biochemical signalling with memory enhancement. *Trans. Comput. Syst. Biology* **11**, 200–215 (2009)
15. J. Jack, A. Rodríguez-Patón, A. Păun, Discrete nondeterministic modelling of Fas pathway. *Int. J. Found. Comput. Sci.* **19**, 1147–1162 (2008)
16. M. A. Colomer, A. Margalida, M.J. Pérez-Jiménez, Population dynamics P system (PDP) models: a standardized protocol for describing and applying novel bio-inspired computing tools. *PLOS ONE* **8**, 1–13 (2013)
17. V. Manca, *Infobiotics: Information in Biotic Systems* (Springer-Verlag, Berlin, 2013)
18. V. Manca, L. Marchetti, Solving dynamical inverse problems by means of MP systems. *BioSyst.* **109**, 78–96 (2012)
19. J. Smaldon, F. J. Romero-Campero, F. Fernandez-Trillo, M. Gheorghe, C. Alexander, N. Krasnogor, A computational study of liposome logic: towards cellular computing from the bottom up. *Syst. Synth. Biology* **4**, 157–179 (2010)
20. S. Adra, T. Sun, S. MacNeil, M. Holcombe, R. Smallwood, Development of a three dimensional multiscale computational model of human epidermis. *PLOS ONE* **5** (2010)
21. S. Sedwards, T. Mazza, Cyto-sim: a formal language model and stochastic simulator of membrane-enclosed biochemical processes. *Bioinform.* **23**, 2800–2802 (2007)
22. F. Ipate, M. Gheorghe, R. Lefticaru, Test generation for P systems using model checking. *J. Logic Algebraic Program.* **79**, 350–362 (2010)
23. C. Jegourel, A. Legay, S. Sedwards, A platform for high performance statistical model checking—plasma. In *Tools and algorithms for the construction and analysis of systems (TACAS’12)* (LNCS Vol. 7214, Berlin 2012), pp. 498–503

24. J. Blakes, J. Twycross, F.J. Romero-Campero, N. Krasnogor, The infobiotics workbench: an integrated in silico modelling platform for systems and synthetic biology. *Bioinform.* **27**, 3323–3324 (2011)
25. J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Simulation of P systems with active membranes on CUDA. *Briefings Bioinform.* **11**, 313–322 (2010)

# Contents

<b>1 Infobiotics Workbench: A P Systems Based Tool for Systems and Synthetic Biology</b>	<b>1</b>
Jonathan Blakes, Jamie Twycross, Savas Konur, Francisco Jose Romero-Campero, Natalio Krasnogor and Marian Gheorghe	
1.1 Introduction	1
1.2 Overview	2
1.2.1 Mathematical Continuous Models	3
1.2.2 Stochastic Discrete Models	3
1.2.3 Executable Modeling Formalisms	4
1.3 Lattice Population P Systems	8
1.4 Infobiotics Workbench	14
1.4.1 Modelling in LPP Systems	16
1.4.2 Simulation	16
1.4.3 Model Checking	20
1.4.4 Optimisation	22
1.5 Case Study	25
1.5.1 LPP Model	26
1.5.2 Simulations	27
1.5.3 Model Checking	28
1.5.4 Supplementary Material	32
1.6 Discussions and Conclusions	33
References	36
<b>2 Statistical Model Checking of Membrane Systems with Peripheral Proteins: Quantifying the Role of Estrogen in Cellular Mitosis and DNA Damage</b>	<b>43</b>
Matteo Cavaliere, Tommaso Mazza and Sean Sedwards	
2.1 Membrane Systems with Peripheral Proteins	43
2.1.1 Formal Language Preliminaries	46
2.1.2 Membrane Systems with Peripheral and Integral Proteins	47
2.2 Statistical Model Checking for Membrane Systems with Peripheral Proteins	53



2.2.1	Temporal Logic as a Query Language . . . . .	53
2.3	A Case Study: The Role of Estrogen in Cellular Mitosis and DNA Damage . . . . .	54
2.4	Methodology and Results . . . . .	57
2.5	Conclusions . . . . .	61
	References . . . . .	62
<b>3</b>	<b>Molecular Diffusion and Compartmentalization in Signal Transduction Pathways: An Application of Membrane Systems to the Study of Bacterial Chemotaxis . . . . .</b>	<b>65</b>
	Paolo Cazzaniga, Daniela Besozzi, Dario Pescini and Giancarlo Mauri	
3.1	Introduction . . . . .	66
3.2	A Multivolume Modeling Approach with Membrane Systems. . .	68
3.2.1	Membrane Systems . . . . .	68
3.2.2	$\tau$ -DPP . . . . .	69
3.3	The Modeling of Bacterial Chemotaxis . . . . .	73
3.3.1	Bacterial Chemotaxis. . . . .	74
3.3.2	A Mechanistic Model . . . . .	76
3.3.3	Multivolume Model: Diffusion in a Signal Transduction Pathway . . . . .	79
3.4	Results . . . . .	82
3.4.1	Simulations of $\Upsilon_{SV}$ . . . . .	82
3.4.2	The Interplay Between Stochastic Fluctuations and the Number of Bacterial Flagella in $\Upsilon_{SV}$ . . . . .	86
3.4.3	Simulations of $\Upsilon_{MV}$ . . . . .	90
3.5	Conclusion . . . . .	93
	References . . . . .	94
<b>4</b>	<b>Membrane System-Based Models for Specifying Dynamical Population Systems . . . . .</b>	<b>97</b>
	M. A. Colomer-Cugat, M. García-Quismondo, L. F. Macías-Ramos, M. A. Martínez-del-Amor, I. Pérez-Hurtado, M. J. Pérez-Jiménez, A. Riscos-Núñez and L. Valencia-Cabrera	
4.1	Introduction . . . . .	98
4.2	Preliminaries. . . . .	100
4.3	A P Systems-Based Probabilistic Modelling Framework. . . . .	100
4.3.1	PDP Systems . . . . .	101
4.3.2	Additional Definitions . . . . .	103
4.3.3	Some Properties of PDP Systems Models . . . . .	104
4.4	An Inference Engine: The DCBA Algorithm . . . . .	105
4.5	Simulation of PDP Systems . . . . .	110
4.5.1	P-Lingua, and the pLinguaCore Library. . . . .	110
4.5.2	The Visual Environment MeCoSim. . . . .	112

4.5.3	Accelerating PDP Systems Simulations . . . . .	113
4.6	A Case Study: Pandemics. . . . .	113
4.6.1	Design of a PDP Modelling Pandemics . . . . .	114
4.6.2	Results. . . . .	125
4.7	Conclusions and Perspectives . . . . .	129
	References . . . . .	131
<b>5</b>	<b>Membrane Systems and Tools Combining Dynamical Structures with Reaction Kinetics for Applications in Chronobiology . . . . .</b>	<b>133</b>
	Thomas Hinze, Jörn Behre, Christian Bodenstein, Gabi Escuela, Gerd Grünert, Petra Hofstedt, Peter Sauer, Sikander Hayat and Peter Dittrich	
5.1	Introduction . . . . .	134
5.2	The KaiABC Core Oscillator: A Circadian Clock Component with Dynamical Molecular Structures . . . . .	135
5.2.1	Biological Background . . . . .	135
5.2.2	Membrane Systems $\Pi_{CSM}$ for Cell Signalling Modules. . . . .	136
5.2.3	Applying $\Pi_{CSM}$ to a KaiABC Core Oscillator Model . . . . .	143
5.2.4	Simulation Case Study. . . . .	145
5.3	Circadian Clocks as Generalised Frequency Control Systems . . . . .	147
5.3.1	A Controllable Goodwin-Type Core Oscillator . . . . .	147
5.3.2	Chemical Frequency Control by Phase-Locked Loops . . . . .	149
5.3.3	Exploring Circadian Clock's Entrainment Behaviour by Simulation Studies . . . . .	153
5.4	Cell Signalling and Gene Regulatory Networks: Logic Circuits in Chronobiological Information Processing . . . . .	157
5.4.1	The General Principle of Cell Signalling in vivo . . . . .	158
5.4.2	Modelling a Bistable Toggle Switch by a Gene Regulatory Network with Two Feedback Loops . . . . .	160
5.4.3	In Vivo Implementation of a Bistable Toggle Switch . . . . .	162
5.5	Spatial Rule-Based Simulator Software SRSim at a Glance . . . . .	164
5.6	Envisioning an Analysis of Membrane System's Static and Dynamical Behaviour by a Constrained-Based Approach . . . . .	168
5.7	Conclusions . . . . .	170
	References . . . . .	170
<b>6</b>	<b>Biochemical Networks Discrete Modeling Inspired by Membrane Systems . . . . .</b>	<b>175</b>
	John Jack, Andrei Păun and Mihaela Păun	
6.1	Introduction . . . . .	176
6.1.1	Modeling with Differential Equations . . . . .	177
6.1.2	Stochastic Methods and the Gillespie Algorithm. . . . .	178
6.1.3	Improving the Gillespie Algorithm . . . . .	179

6.1.4	Our Work . . . . .	180
6.2	Membrane Systems as Cell Simulators . . . . .	180
6.2.1	Description of the NWT Algorithm . . . . .	182
6.2.2	Maintaining the Min-heap . . . . .	184
6.2.3	Memory Enhancement . . . . .	185
6.2.4	Case 1: Deterministic Memory Enhancement . . . . .	186
6.2.5	Case 2: Nondeterministic Memory Enhancement . . . . .	188
6.3	Comparing the NWT Algorithm with the ODE and Gillespie's Algorithm . . . . .	191
6.4	Modeling FAS-Induced Apoptosis . . . . .	194
6.4.1	Apoptotic Signaling Cascades . . . . .	195
6.4.2	Fas-Mediated Apoptosis . . . . .	195
6.4.3	Results of Discrete Method . . . . .	197
6.4.4	Bcl-2's Effects on the Type II Pathway . . . . .	198
6.4.5	Modeling the Behavior of the Type I Pathway . . . . .	200
6.4.6	Summary for the FAS Simulation . . . . .	201
6.5	HIV-1 Effects on the FAS Pathway . . . . .	203
6.5.1	A Brief History of HIV . . . . .	203
6.5.2	AIDS Pathogenesis . . . . .	204
6.5.3	HIV-1 Infection . . . . .	205
6.5.4	HIV-1-Related Effects on the Fas Pathway . . . . .	208
6.5.5	Modeling Results . . . . .	208
6.5.6	Summary for Simulating HIV Latency . . . . .	213
6.6	Conclusions and Final Remarks . . . . .	214
6.6.1	Extensions on the HIV Model . . . . .	214
6.6.2	Calcium's Role in Apoptosis . . . . .	216
	References . . . . .	216
<b>7</b>	<b>MP Modelling for Systems Biology: Two Case Studies . . . . .</b>	<b>223</b>
	Luca Marchetti, Vincenzo Manca, Roberto Pagliarini and Aliccia Bollig-Fischer	
7.1	Introduction . . . . .	223
7.1.1	Log-Gain Stoichiometric Stepwise Regression (LGSS) . . . . .	226
7.2	The Glucose/Insulin Dynamics in the Intravenous Glucose Tolerance Test (IVGTT) . . . . .	228
7.2.1	Mathematical Models of the Intravenous Glucose Tolerance Test . . . . .	230
7.2.2	MP Modelling of IVGTT . . . . .	231
7.3	MP Modelling of Gene Networks . . . . .	236
7.3.1	From Raw Data to MP Models . . . . .	239
7.4	Conclusion and Ongoing Research . . . . .	242
	References . . . . .	243

<b>8</b>	<b>Modelling and Analysis of <i>E. coli</i> Respiratory Chain.</b>	<b>247</b>
	Adrian Țurcanu, Laurențiu Mierlă, Florentin Ipate, Alin Stefanescu, Hao Bai, Mike Holcombe and Simon Coakley	
8.1	Introduction	248
8.2	Background	249
8.2.1	kP Systems.	249
8.2.2	X-Machines	252
8.3	General Description of <i>E. coli</i> .	252
8.4	FLAME Simulations of <i>E. coli</i> Respiratory Chain.	254
8.5	A Kernel P System Corresponding to <i>E. coli</i>	257
8.6	Modelling, Simulation and Verification	258
8.6.1	Implementation in Event-B for ProB	258
8.6.2	Implementation in Promela for Spin	260
8.6.2	Implementation in Promela for Spin	260
8.6.3	Simulation Results	261
8.6.4	Verification Results.	262
8.6.5	Event-B Versus Promela	264
8.7	Conclusions	264
	References	265

# Chapter 1

## Infobiotics Workbench: A P Systems Based Tool for Systems and Synthetic Biology

Jonathan Blakes, Jamie Twycross, Savas Konur, Francisco Jose Romero-Campero, Natalio Krasnogor and Marian Gheorghe

**Abstract** This chapter gives an overview of an integrated software suite, the Infobiotics Workbench, which is based on a novel spatial discrete-stochastic P systems modelling framework. The Workbench incorporates three important features, simulation, model checking and optimisation. Its capability for building, analysing and optimising large spatially discrete and stochastic models of multi-cellular systems makes it a useful, coherent and comprehensive in silico tool in systems and synthetic biology research.

### 1.1 Introduction

Membrane computing is a growing area of research in computer science and, more specifically, natural computation. Membrane computing assumes that the processes taking place in the compartments of a living cell can be interpreted as computations.

---

J. Blakes (✉) · J. Twycross · N. Krasnogor  
ICOS Research Group, School of Computer Science, University of Nottingham, Nottingham, UK  
e-mail: jonathan.blakes@nottingham.ac.uk

J. Twycross  
e-mail: jamie.twycross@nottingham.ac.uk

N. Krasnogor  
e-mail: natalio.krasnogor@nottingham.ac.uk

S. Konur · M. Gheorghe  
Department of Computer Science, University of Sheffield, Sheffield, UK  
e-mail: s.konur@sheffield.ac.uk

M. Gheorghe  
e-mail: m.gheorghe@sheffield.ac.uk

F. J. Romero-Campero  
Department of Computer Science and Artificial Intelligence, University of Seville, Seville, Spain  
e-mail: fran@us.es

The devices of this model are called P systems. A P system consists of a cell-like membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules. Because a set of rules is a mathematical entity, it can be analysed with formal rigour to discover the relationships between rules and their subjects, potential sequences of events, and the reachability of certain states.

The Infobiotics Workbench is an integrated *stochastic P systems* based platform for computer-aided modelling, design and analysis of large-scale biological systems which consists of three key components: (a) a *simulator for a modelling language*—discussed in Sect. 1.4.2; (b) a *model checking* module—see Sect. 1.4.3; and (c) a *model structure and parameter optimisation* engine—details in Sect. 1.4.4. The availability of deterministic and multi-compartment stochastic simulation of population models enables comparisons between macroscopic and mesoscopic interpretations of molecular interaction networks and investigation of temporo-spatial phenomena in multicellular systems. Model checking can be used to increase confidence in simulated observations by quantifying the probability of reaching definable states for all possible trajectories [75]. The optimisation component of the Workbench enables designs of synthetic circuits matching a set of desired temporal dynamics (specified as time series of molecular species quantities) to be automatically composed from modules of abstract networks motifs and/or completely specified bioparts (with corresponding DNA sequences) drawn from libraries of reusable model components.

The modelling language allows specifications of cellular populations distributed over different geometric surfaces, like lattices. The simulation results capabilities of the Infobiotics Workbench enables molecular populations to be animated as a surface over the cellular population for a visually rich semi-quantitative analysis of behaviour in space as well as time. Time series of molecular quantities (as concentrations or number of molecules) in individual or averaged simulation runs can be plotted for any combination of species, compartments and timepoints, enabling a fine-grained quantitative comparison of expected and simulated temporal dynamics at multiple locations in spatial models. Histograms are used to estimate the distributions of molecular species across cellular components or runs at different timepoints, possibly revealing differentiation of cell states as initially homogeneous populations diverge through emergent behaviours arising from the (stochastic) application of reaction rules.

This chapter is divided into the following sections: an overview of various formalisms used in modelling biological systems; a presentation of the lattice population P systems; a description of the key components of the Infobiotics Workbench; a case study; and finally discussions regarding the benefits of the modelling framework presented over other similar approaches and future developments.

## 1.2 Overview

In this section, we give an overview of established and emerging mathematical and computational formalisms used to model biological systems.

### 1.2.1 Mathematical Continuous Models

The vast majority of models used in systems biology have, until recently, been mathematical, based on systems of coupled ordinary differential equations (ODEs). In an ODE model each molecular species in the model is defined as a single variable which represents its concentration over time. The correctness of an ODE model relies on the assumption that concentrations vary (with respect to time) *continuously* and *deterministically*. ODEs aim to approximate the stochastic process, but actually represent the limit of the stochastic process as the number of molecules and volume are taken to infinity while maintaining their ratio constant. This assumption is only valid when the number of molecules is sufficiently high (an approximate lower bound is  $10^3$  molecules) and reactions are fast.

### 1.2.2 Stochastic Discrete Models

When the number of particles of the reacting species is small and reactions are slow, as is frequently the case for genetic regulation in biological systems, the previous assumption is questionable and the deterministic continuous approach to chemical kinetics should be complemented by an alternative approach. In this respect, one has to recognise that the individual chemical reaction steps occur discretely and are separated by time intervals of random length. *Discrete* and *stochastic* approaches are more accurate in this situation, and these mechanistic formulations also have the advantage of being closer to the molecular biological interactions that constitute our understanding. Stochastic models are apparently closer to the underlying model on which ODEs are based (the CME) and may produce behaviour that is more typical of real systems.

In a discrete species population model of a chemical system, the state of the system is defined by the number of molecules of each chemical species at any given time. The *Chemical Master Equation* (CME) completely determines the probabilities of each reaction in a well-mixed chemical system, at constant temperature and volume, given the current state. The assumption of well-mixed systems allows the analysis to consider populations (multisets) of molecules, rather than individual molecules with spatial positions, and thus use a single rate constant for mass action kinetics.

The CME represents a continuous-time Markov chain which can capture the noise (stochasticity) in the system. Unfortunately the CME is actually a system of as many coupled ordinary differential equations as there are combinations of molecules that can exist in the system, and can only be solved analytically for a very few simple systems [58]. Fortunately a more tractable approach exists. Instead of solving the CME we can construct numerical realisations of the system's state over time, that is, generate trajectories of the system using a kinetic Monte Carlo algorithm, Gillespie's *stochastic simulation algorithm* (SSA) [55], in *exact* compliance with the CME.

Gillespie initially produced two SSAs that simulate every reaction in the system: the First Reaction Method [54] and the simpler but equivalent Direct Method [55]; and subsequently showed these to be a rigorous derivation of the CME [56]. More efficient exact SSAs have been introduced since, including the dominant Next Reaction Method (NRM) [51] which scales logarithmically with the number of reactions, the Next Subvolume Method [38] as a variation on NRM for discrete-space intracellular models, the Partial-propensity Direct Method [106] scaling at most linearly with the number of species (often far fewer than reactions), and the Composition-Rejection SSA [117] offering constant-time performance for  $10^5$  or greater reactions. *Approximate* methods, that simulate batches of fast non-critical reactions, include  $\tau$ -leaping (established in Ref. [57] and optimised in Ref. [21]) and the slow-scale SSA [22]. These offer accelerated performance for stiff systems, with an acceptable and tunable loss of accuracy, and enable larger models to be simulated in reasonable time.

### 1.2.3 Executable Modeling Formalisms

The formalisation of biological systems using alternatives to mathematical equations has recently received much interest as a deeper mechanistic understanding of biological systems is sought through modelling. Formalisms where molecular populations and interactions are modelled as discrete entities and events have come to be known collectively as *Executable Biology*. *Executable biology* [42, 43], or *algorithmic systems biology* [102], propose the application of established computational formalisms from other domains, and domain specific languages for the formalisation and implementation of biological models. Below we review a selection of these alternative representations, their capabilities and implementations.

**The Systems Biology Markup Language (SBML)** [69] is an XML dialect used to store and exchange models of biological systems between different tools. SBML files store information about model compartments, species and reactions, as well as events, units, etc. that are relevant to some models and approaches but not others. Tools for the visual specification of models in SBML, e.g. *CellDesigner* [47], *e-cell* [125], *VCell* [87] and *COPASI* [68], enable the visual creation of models from a collection of symbols for various types of molecular and interactions.

**Cellular automata** were studied in the early 1950s as a possible model for biological systems ([124], p. 48). This formalism, inspired from cellular biology, has been extensively used in modelling a broad spectrum of biological systems, amongst them pattern formation (morphogenesis) [31], ecology and population biology, immunology, oscillations, diffusion processes, fibroblast aggregation, ant trails and others (for more details see the overview paper [39]). In the paper coining the term algorithmic systems biology, cellular automata are mentioned amongst the models employing explicitly computational aspects [102].

Cellular automata have been connected to membrane systems for different modelling reasons. In Ref. [26] it is studied the behaviour of HIV infection by



comparing a cellular automaton model and a conform-P system model with respect to the robustness related to various initial conditions and parameters. The possibility of converting a cellular automaton into a generalised P systems has been also investigated [89].

**Boolean networks** [72] are one of the oldest examples of executable biological modelling formalisms. They represent the interactions of genes as a directed graph. Each node of a Boolean network can represent a gene that is either active, or inactive. Edges between nodes contribute either positively (activation) or negatively (inactivation) to the node at which they are directed (providing the node from which the edge extends is active), modelling hierarchies of genetic regulations. Boolean networks are deterministic given their starting configuration for which there are  $2^n$  possible system-wide states where  $n$  is the number of nodes.

Boolean networks are qualitative in terms of quantities and time. With only topological data and binary relationships required to build a model, Boolean networks can usually be constructed when data is scarce, and are therefore often chosen as a modelling formalism for their amenability to analysis rather than realism [41].

Similarly qualitative but more fine-grained are **Statecharts**, a method devised for the engineering of complex reactive systems. Statecharts have been used to successfully model the interactions of two signalling pathways, specifying the fates of the six vulval precursor cells, which provide a mechanism for pattern formation during the *C.elegans* development [44].

**Petri nets** are formalisms that model systems with concurrent behaviour and are particularly suited to modelling discrete asynchronous distributed systems. Petri nets were initially applied to biological pathways [107, 108] for semi-quantitative analysis in terms of discrete number of objects and uniform time intervals. A bibliography [123] of Petri nets applications in biomolecular modelling, simulation and analysis summarises developments up to 2002. More recent contributions include the ubiquitously studied ERK signal transduction pathway [53], receptor signalling and kinase cascades, cell-cycle regulation and wound healing [52], and synthetic biology [65].

A quantitative notion of time is introduced by *stochastic Petri nets* [88, 120], where each transition has an associated rate from which a period of time is calculated upon firing and added to the global clock, typically using a stochastic simulation algorithm. *Coloured Petri nets* [71] can provide a novel way of dealing with the combinatorial explosion of states, where differently coloured tokens can represent molecules of the place's species with various modifications, or alternatively molecules in different cells without extrapolating the Petri net [49].

There are numerous tools deployed to create and analyse Petri nets. We refer the reader to Ref. [97] for the database of the available tools.

**Process algebras** (or process calculi) are a diverse family of related formalisms that describe distributed concurrent processes, such as the objects inside a computer program or a collection of programs, interacting.  $\pi$ -calculus [86], for concurrent mobile processes, is an accepted model for interacting systems with communication topologies that evolve dynamically [92]. For biological models, process algebras consider molecules with binding sites as processes with communication channels.

In standard  $\pi$ -calculus the system evolves in uniform time steps with each communication being equally likely, irrespective of the number of channels; such a simulation is semi-quantitative in the same way as a standard Petri net.

*Stochastic  $\pi$ -calculus* (initially proposed as  $S\pi$  [101]) enables fully quantitative simulations by associating a rate constant with each channel. BioSPI [104], the first stochastic  $\pi$ -calculus simulator [109], could simulate systems with hundreds of processes in the order of seconds [103]. The current leading implementation of a stochastic  $\pi$ -calculus simulator is SPiM [98]. A more intuitive understanding of  $\pi$ -calculus is made possible by a graphical representation [99] that visualises the state-space of each process as a graph and has been incorporated into SPiM. In Ref. [99] a graphical execution model was defined and proved equivalent to  $S\pi$ .

*Performance Evaluation Process Algebra (PEPA)* is an alternative stochastic process algebra that has been applied to modelling signalling pathways [15–18] and synthetic biology designs [50]. PEPA can be used for reagent-centric and pathway-centric modelling [17]. *Bio-PEPA* [25] is a biologically-oriented modification of PEPA incorporating stoichiometry and the use of kinetic laws in rate functions.

*BlenX* [30] is a high level textual language grounded in process algebra, explicitly designed to model biological entities and their interactions, providing several features not found up until now in stochastic process algebras. For example, it uses a *type* file which specifies stochastic rates between interacting types rather than embedding those rates into the model as stochastic constants. BlenX is supported by a set of tools collectively known as Beta Workbench [29] including a graphical model editor, stochastic simulator and a plotter for displaying model execution time courses. A unique feature of the plotter is the ability to plot causality, where each simulation event (molecular interaction) is drawn as a box inside the box of the event that led to it. Other prototype tools being developed to support BlenX include KInfer which performs model and kinetics inference by estimating reactions and rate constants from real concentration data measured at discrete time points.

We refer the reader to Ref. [61] for an extensive review on the application of process algebras to biological modelling up to 2006. Other notable works include GEC [93] and LCS [94].

**Membrane computing** [91] is a branch of natural computing that emphasises the compartmentalised nature of biological systems and its power in computation. The central objects are P systems, that consist of a membrane structure, the regions of which contain rewriting rules operating on multisets of objects [112]. The P system *evolves* by the repeated application of rules, mimicking chemical reactions and transportation across membranes, and halts when no more rules can be applied.

The closeness of this representation to the biology make P systems highly suited as a communication device between computer scientists and biologists collaborating on a model. Some of the most well-studied P systems with relevance for modelling biological systems are presented below.

- *Deterministic and non-deterministic P systems* consisting of a broad range of models:

- *Metabolic P systems* (MP systems) have diverged considerably from the non-deterministic, compartmentation-based notion of P systems, being coarse-grained models of the fluxes between molecular populations within a single membrane computed by means of the metabolic algorithm (MA)—its equational formulation is in Ref. [79]. A methodology for inferring and validating the model has been elaborated [82]. An overall presentation of these systems is available in Ref. [80] and a comprehensive description in [81]. MP systems are supported by the MetaPlab [85] software, previously Psim [11].
- *Non-deterministic P systems* are used in a context where the rules are selected according to a waiting time algorithm involving a mass action law principle [70]; this model is successfully utilised to analyse the behaviour of different biochemical signalling networks. Another special class of P systems, called *conformon-P systems*, deals with systems having rewriting and communication rules using together with multisets, some numerical values that help controlling the computation. These models have been used to study how some diseases spread [26].
- *Probabilistic (stochastic) P systems* include several classes of P systems:
  - *Stochastic P systems* (SP systems) [114] directly apply stochastic rate constants and Gillespie’s stochastic simulation algorithms to P systems, with boundary rules that make the specification of molecule transport between enclosed and enclosing members simple and intuitive. These are discussed in much greater detail in Sect. 1.3.
  - *Dynamical Probabilistic P systems* (DPP) [96] use standard P systems with a novel rule application method to model biological phenomena in a discrete and stochastic way (motivated by the investigation of maximal parallelism in nature). In a procedure not unlike propensity calculation in the Gillespie algorithm DPP rules are dynamically assigned a probability that is the product of the possible combinations of reactant objects and an associated rate. A tau-leaping variant of it is also provided [24] which is packaged in the BioSimWare software platform [7].
  - *Probabilistic Dynamics Population P systems* represent a class of P systems meant to provide an accurate model of multi-environmental systems; it has applications to ecosystems, where the methodology consists of a modular specification including probabilistic rules [28] describing transformations within compartments as well as communications between compartments and cooperations involving different parts of the environment. This approach is included in the P-Lingua framework [115] and has a number of implementations, including one that uses GPU hardware [83]. An integrated software environment, called MeCoSim, is supporting the modelling language with an editor and different visualisation options [95].
  - *Probabilistic P systems* with peripheral proteins focuses on trans-membrane operations where a Gillespie algorithm is used for describing the system behaviour; a specification language is integrated into the simulation environment Cyto-sim [23].

- *Extension of P systems with string objects* for modelling protein binding domains with ligands have been considered for specifying oscillatory phenomena; a software environment, called SRSim, which incorporates spatial rules and a strong visualisation engine is available [67].

In this volume some of the above mentioned variants of P systems, like metabolic P systems, non-deterministic P systems, dynamical probabilistic P systems, probabilistic dynamics P systems and probabilistic P systems with peripheral proteins, appear as models of various biological systems or scenarios.

A *general-purpose class of computational tools* has been introduced for tackling the challenge of a combinatorial explosion in the number of interactions that arises when many species with coincidental modifications, conformations or states need to be represented explicitly. Some of the most prominent rule-based systems that deal with these issues are NFsim [118], BioNetGen [40], Kappa [27] and little b [78]. While each of these approaches can model some aspects regarding pathways and their molecular components, none of the approaches can fully capture “quantitative dynamics, interactions among molecular entities and structural organisation of cells” [112].

### 1.3 Lattice Population P Systems

Many multicellular biological systems have a spatial component where molecule exchange between *adjacent* cells determines the overall phenotypes. However, this structure cannot be captured by stochastic P systems, which have only a hierarchical membrane structure of compartments within other compartments or a simple population of such entities. Therefore, stochastic P systems need to be augmented with an additional level of organisation, a 2D geometric lattice on which a population of P systems can be placed and over which molecules can be translocated. Rules that move objects from one P system to another on the lattice are associated a vector that describes where to put that molecules. We call this extension of stochastic P systems *Lattice Population P systems (LPP systems* for short) and, in the tradition of P systems, proceed with their formal definition (published in Ref. [114]).

Each cell type with its compartmentalised structure, characteristic molecular species and molecular processes, is represented using a *stochastic system* according to Definition 1. The rules of each such system are possibly specified in a modular way. The spatial distribution of cells in the population is represented using a *finite point lattice*, Definition 2, and finally different copies of the corresponding stochastic system representing each cell type are distributed over the points of the lattice according to the spatial distribution of an *LPP systems* in Definition 3.

Before providing the formal definitions mentioned above let us notice that the idea of a lattice of functional units has been discussed for conformon-P systems [26] and stochastic P systems distributed in communicating environments [9] have been studied.

**Definition 1** A **stochastic P system (SP system)** is a formal rule-based specification of a multicompartmental and discrete dynamical system with stochastic semantics given by a tuple:

$$SP = (O, L, \mu, M_1, \dots, M_n, R_1, \dots, R_n) \quad (1.1)$$

where:

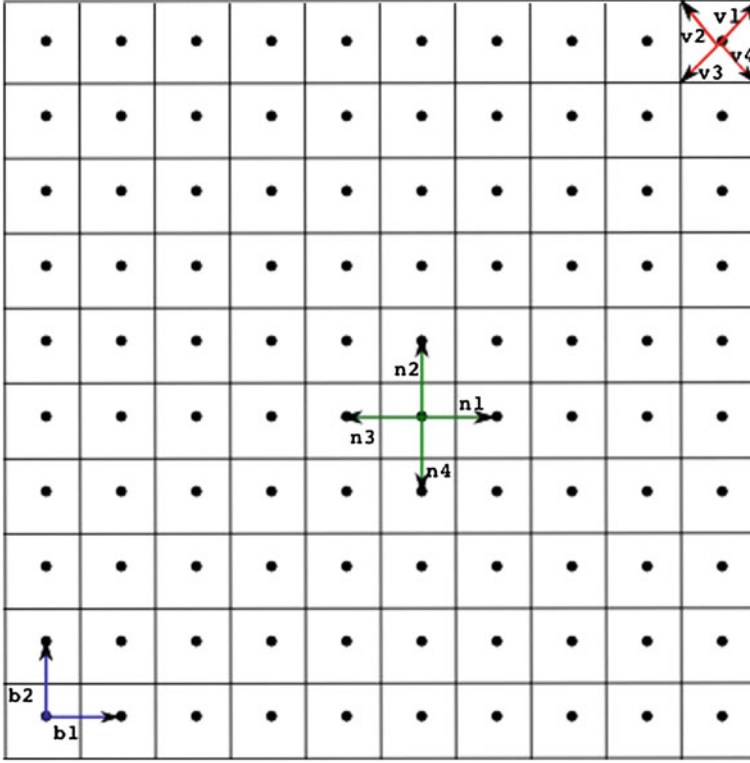
- $O$  is a finite set (alphabet) of objects specifying the entities involved in the system (genes, RNAs, proteins, etc.);
- $L = \{l_1, \dots, l_n\}$  is a finite set of labels naming compartments (e.g. nucleus);
- $\mu$  is membrane structure composed of  $n \geq 1$  membranes defining the regions or compartments of the system. The outermost membrane is called the *skin membrane*;
- $M_i = (l_i, w_i, s_i)$ , for each  $1 \leq i \leq n$ , is the initial configuration of the compartment or region defined by the membrane  $i$ , where  $l_i \in L$  is the label of the membrane,  $w_i \in O^*$  is a finite multiset of objects and  $s_i$  is a finite set of strings over  $O$  (in this presentation the strings will not be used);
- $R_{l_k} = \{r_1^{l_k}, \dots, r_{m_{l_k}}^{l_k}\}$ , for each  $1 \leq k \leq n$  is a set of multiset rewriting rules describing the interactions between the molecules, such as complex formation and gene regulation. Each set of rewriting rules  $R_{l_k}$  is specifically associated to the compartment identified by the label  $l_k$ . These multiset rewriting rules are of the following form:

$$r_i^{l_k} : o_1 [o_2]_l \xrightarrow{c_i^{l_k}} o'_1 [o'_2]_l \quad (1.2)$$

where  $o_1, o_2$  and  $o'_1, o'_2$  are multisets of objects (possibly empty), over  $O$ , representing the molecular species consumed and produced in the corresponding molecular interaction. The square brackets and the label  $l$  describe the compartment involved in the interaction. An application of a rule of this form changes the content of the membrane with label  $l$  by replacing the multisite  $o_2$  with  $o'_2$  and the content of the membrane outside by replacing the objects  $o_1$  with  $o'_1$ . The stochastic constant  $c_i^{l_k}$  is used to compute the propensity of the rule by multiplying it by the number of available reactants in the membrane, where the same object is not counted twice for homogenous bimolecular reactions [6]. The propensity associated with each rule is used to compute the probability and time needed to apply it (according to the stochastic semantics of Gillespie's theory of chemical kinetics [54]).

Definition 1 provides the formalism needed for the specification of an individual cell with its structure given by  $\mu$  and the outer membrane called the *skin membrane*. To specify the possible spatial distribution of cells assembled into colonies and tissues we define an array of regularly distributed points according to a *finite point lattice* or *grid* [77] capable of describing the spatial geometries (see Fig. 1.1).

This model looks very similar to a cellular automaton although in lattice population P systems we have considered that each cell of the grid has a cell-like stochastic membrane system inside and this type of grid has been chosen to illustrate a specific geometry we have considered so far. Our model is more general than a cellular



**Fig. 1.1** A square lattice

automaton and in the future some more complex geometries describing 3D complex structures will be introduced.

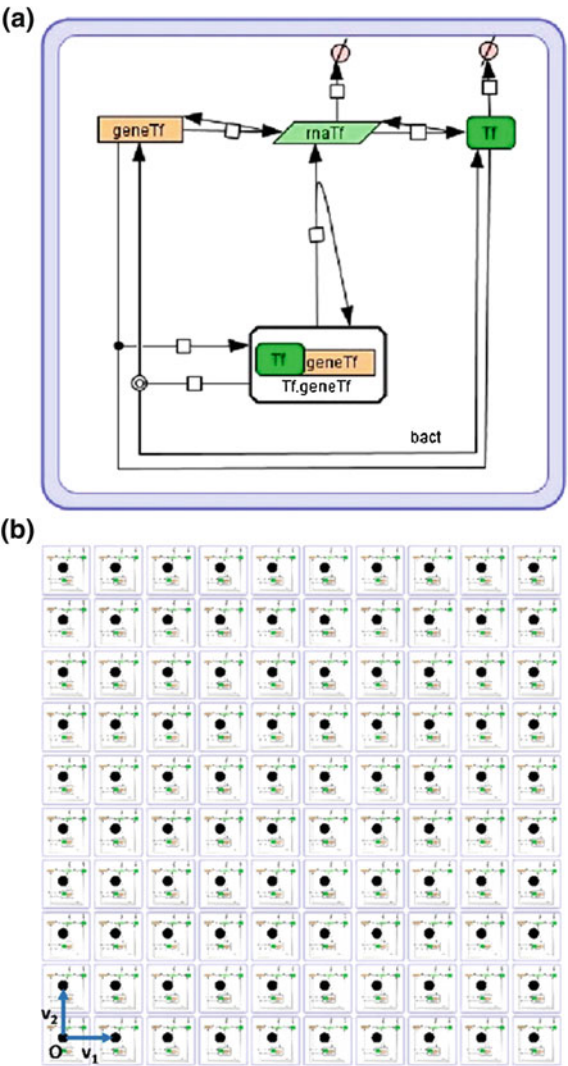
**Definition 2** Given  $B = \{v_1, \dots, v_n\}$  a list of linearly independent basis vectors,  $o \in \mathbb{R}^n$  a point referred to as origin and a list of integer bounds  $(\alpha_1^{min}, \alpha_1^{max}, \dots, \alpha_n^{min}, \alpha_n^{max})$ , a **finite point lattice** generated by:

$$Lat = (B, o, (\alpha_1^{min}, \alpha_1^{max}, \dots, \alpha_n^{min}, \alpha_n^{max})) \quad (1.3)$$

is the collection of regularly distributed points,  $P(Lat)$ , obtained as follows:

$$P(Lat) = \{o + \sum_{i=1}^n \alpha_i v_i : \forall i = 1, \dots, n (\alpha_i \in \mathbb{Z} \wedge \alpha_i^{min} \leq \alpha_i \leq \alpha_i^{max})\} \quad (1.4)$$

Given a *finite point lattice*, generated by  $Lat$ , each point  $x = o + \sum_{i=1}^n \alpha_i v_i \in P(Lat)$  is uniquely identified by the coefficients  $\{\alpha_i : i = 1, \dots, n\}$  and consequently it will be denoted as  $x = (\alpha_1, \dots, \alpha_n)$ .



**Fig. 1.2** SP systems containing reactions of a gene network, single (a) and distributed over the LPP system lattice (b)

SP systems are distributed on the lattice according to an LPP system (see Definition 3), as shown in Fig. 1.2.

**Definition 3** A **lattice population P system**, or **LPP system** for short, is a formal specification of an ensemble of cells distributed according to a specific geometric disposition given by the following tuple:



$$LPP = (Lat, \{SP_1, \dots, SP_p\}, Pos, \{T_1, \dots, T_p\}) \quad (1.5)$$

where

- $Lat$  defines a finite point lattice in  $\mathbb{R}^n$  (typically  $n = 2$ ) as in Definition 2 that describes the geometry of cellular population.
- $SP_1, \dots, SP_p$  are SP systems as in Definition 1 specifying the different cell types in the population.
- $Pos : P(Lat) \rightarrow \{SP_1, \dots, SP_p\}$  is a function distributing different copies of the SP systems  $SP_1, \dots, SP_p$  over the points of the lattice.
- $T_k = \{r_1^k, \dots, r_{n_k}^k\}$  for each  $1 \leq k \leq p$  is a finite set of rewriting rules termed *translocation rules* that are added to the skin membrane of the respective SP system  $SP_k$  in order to allow the interchange of objects between SP systems located in different points in the lattice. These rules are of the following form:

$$r_i^k : [obj]_k \bowtie [\mathbf{v}]_{k'} \xrightarrow{c_i^k} [ ]_k \bowtie [obj]_{k'} \quad (1.6)$$

where  $obj$  is a multiset of objects,  $\mathbf{v}$  is a vector in  $\mathbb{R}^n$  and  $c_i^k$  is the stochastic constant used in our algorithm to determine the dynamics of rule applications. The application of a rule of this form in the skin membrane with the label  $l$  of the SP system  $SP_k$  located in the point  $\mathbf{p}$ ,  $Pos(\mathbf{p}) = SP_k$ , removes the objects  $obj$  from this membrane and places them in the skin membrane of the SP system  $SP_{k'}$  located at the point  $\mathbf{p} + \mathbf{v}$ ,  $Pos(\mathbf{p} + \mathbf{v}) = SP_{k'}$ . Note that vectors allow for any topology to be encoded in the lattice geometry.

Molecular reaction networks can, to a certain degree, be decomposed into modules acting as discrete entities carrying out particular tasks [64]. It has been shown that there exist specific modules termed *motifs* that appear recurrently in transcriptional networks performing specific functions like response acceleration and noise filtering [1]. Modularisation is also a central technique used in the engineering of synthetic cellular systems by combining well-characterised and standardised cellular models [19] as exemplified in the MIT BioBricks project [116].

Definition 4 gives the definition of a *P system module* that we use [113] to decompose large sets of rules into more meaningful and reusable subsets. Other similar concepts of modularity in P systems for various other classes of P systems. Modules of a conformon-P systems are discussed in Ref. [46]. In [32] P modules are introduced with the aim of facilitating a modular decomposition of complex P systems, whereas in Ref. [66] it is defined as a functional unit fulfilling some elementary computational tasks. In the context of generalised communicating P systems [122] it is introduced a concept of a module as a network of cells. Subsequently we introduce the concept of a module for stochastic P systems with the aim of capturing some high level behaviour which can be characterised by some specific parameters and which outlines some generic names that are instantiated with specific values in various contexts.



**Definition 4** A **P system module**,  $Mod$ , is parameterised with three finite ordered sets of variables  $O = \{O_1, \dots, O_x\}$ ,  $C = \{C_1, \dots, C_y\}$  and  $Lab = \{L_1, \dots, L_z\}$  (objects, stochastic rate constants and compartment labels respectively), and consists of a finite set of rewriting rules of the form in Eq. (1.3):

$$Mod(O, C, Lab) = \{r_1, \dots, r_m\} \quad (1.7)$$

The objects, stochastic constants and labels of the rules in module  $Mod$  can contain variables from  $O$ ,  $C$  or  $Lab$  which are *instantiated* with specific values  $o = \{o_1, \dots, o_x\}$ ,  $c = \{c_1, \dots, c_y\}$  and  $lab = \{l_1, \dots, l_z\}$  for  $O$ ,  $C$  and  $Lab$  respectively as in:

$$Mod(\{o_1, \dots, o_x\}, \{c_1, \dots, c_y\}, \{l_1, \dots, l_z\}) \quad (1.8)$$

the rules are obtained by applying the corresponding substitutions  $O_1 = o_1, \dots, O_x = o_x$ ,  $C_1 = c_1, \dots, C_y = c_y$  and  $L_1 = l_1, \dots, L_z = l_z$ .

Our definition of *P system module* allows the hierarchical description of a complex module,  $M(O, C, Lab)$ , by obtaining its rules as the set union of simpler modules,  $M(O, C, Lab) = M_1(O_1, C_1, Lab_1) \cup \dots \cup M_q(O_q, C_q, Lab_q)$  with  $O = O_1 \cup \dots \cup O_q$ ,  $C = C_1 \cup \dots \cup C_q$  and  $Lab = Lab_1 \cup \dots \cup Lab_q$ .

Finally, the set of rules,  $R_{l_k}$ , in *SP systems* can be specified in a modular way as the set union of several instantiated *P system modules*,  $R_{l_k} = M_1(o_1, c_1, lab_1) \cup \dots \cup M_{q_k}(o_{q_k}, c_{q_k}, lab_{q_k})$ .

The use of modularity allows us to define libraries or collections of modules:

$$Lib = \{Mod_1(O_1, C_1, Lab_1), \dots, Mod_p(O_p, C_p, Lab_p)\} \quad (1.9)$$

An SP system model may contain instantiations of modules from multiple libraries, and the same module can be instantiated multiple times with different parameters. In Sect. 1.5 we provide examples for SP system models, libraries and lattice systems.

P systems modules can be made more or less abstract by changing the number of components exposed as parameters (species identities and stochastic rate constants). Motifs of biological networks, corresponding to the topology of the underlying reaction network modelled at a particular level of detail, can be captured by fully abstract modules where all components are parameters. In this usage the names of parameters should indicate the role that their values will play in the module.

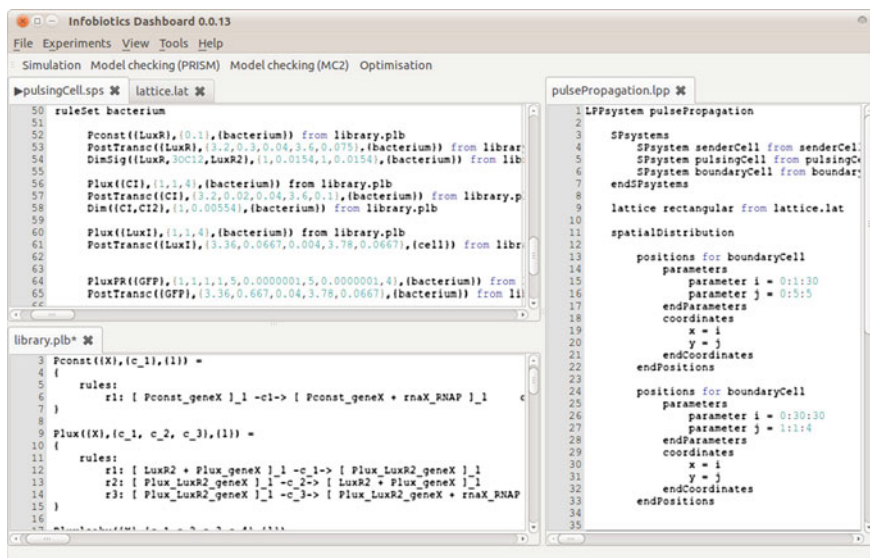
Well-characterised synthetic biological parts and devices can be captured by fully concrete modules (i.e. without parameters) because the identity of every species and the stochastic rate constants of each reaction are validated.

## 1.4 Infobiotics Workbench

The Infobiotics Workbench (IBW)<sup>1</sup> is an integrated software suite of tools to perform *in silico* experiments for LPP models in Systems and Synthetic Biology [14]. Models are simulated either using stochastic simulation or deterministic numerical integration using MCSS, an application for simulating multi-compartment stochastic P system models, and visualised in time and space with the *Infobiotics Dashboard*. Model structure and parameters can be optimised with evolutionary algorithms using OPTIMIZER, and properties of a model's temporo-spatial behaviour calculated using probabilistic or simulative model checking with PMODELCHECKER.

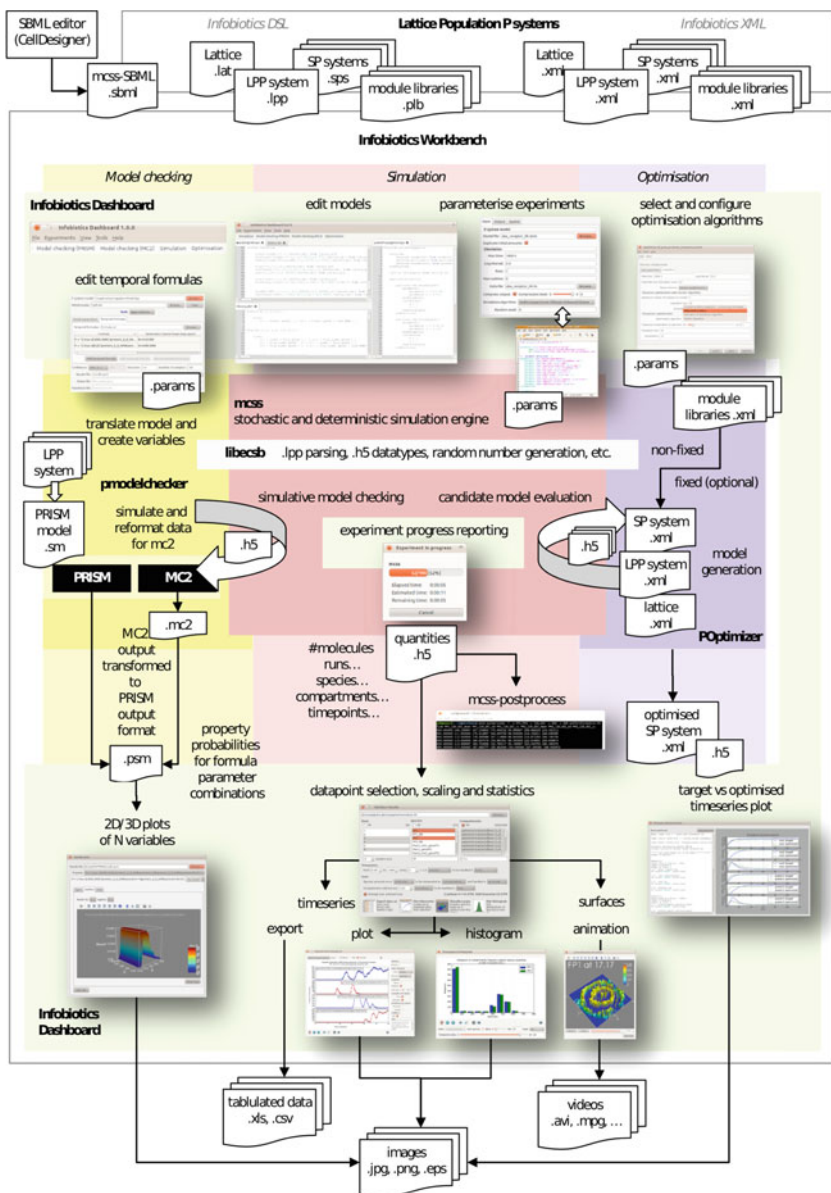
The Infobiotics Dashboard window uses an adjustable tabbed interface to display multiple views on to files (Fig. 1.3). LPP DSL specifications of Infobiotics models can be edited with the simple editor provided by the Dashboard or an external editor of the user's choosing.

In IBW, the *experiments* can be accessed through the integrated interface or with individual GUIs outside the workbench. Experiments are parameterised with XML parameter files, edited interactively with help and validation, and performed within the GUI. Figure 1.4 summarises the overall flow of information through the components of the Infobiotics Workbench.



**Fig. 1.3** The Infobiotics Dashboard with multiple text editors displaying LPP system DSL files for a pulse generating synthetic biology model

<sup>1</sup> <http://www.infobiotics.org>



**Fig. 1.4** Flow of information through the components of the Infobiotics Workbench. Data is passed between components as files. Parameter files (.params), referencing model files (.sbml, .lpp or .xml), are produced by the Infobiotics Dashboard and supplied to the experiment executables for simulation (MCSS), model checking (PMODELCHECKER) and optimisation (POPTIMIZER). Executables communicate progress to `stdout` which is read and interpreted by the Dashboard to report the percentage completed and estimate time remaining. Files produced by the experiments (.h5 simulation data, .psm model checking property probabilities) are presented by the Dashboard for analysis, and can be exported as tabulated data, images and video files

### 1.4.1 Modelling in LPP Systems

For LPP system models to be specified and manipulated by computers it is necessary that they have a machine-readable equivalent. LPP system XML is a set of machine-readable data formats which closely mirrors our formal definitions. It allows us to define, in a single file or multiple files, modules of stochastic P system rules, P systems with initial multisets and instantiations of modules of rules, a geometric lattice and distribution of P systems over the lattice, which together constitute an LPP system model.

The LPP XML formats are well suited to software development with LPP systems, but clearly writing models in XML by hand and reading them back is a cumbersome process with syntax obscuring information. A parser for an **LPP system DSL** (domain-specific language) that is essentially the XML formats without the angle-brackets, quotes and some closing tags has been developed. The parser is used to read DSL files directly, but it also silently converts them into XML.

The LPP formalism enables three types of modelling component reuse:

- *Inter-model reuse*: Modules (in libraries), SP systems and lattices (encoding neighbourhood relationships between SP systems in 2D space) reside in different files which can be referred to by multiple LPP system models.
- *Intra-model reuse*: Multiple copies of different SP system can be placed within each LPP system, facilitating the building of models of homogeneous or heterogeneous bacterial colonies or tissues.
- *Intra-submodel reuse*: Parameterisable modules of rules can be instantiated multiple times within each compartment of an SP system, using different parameters (species identities and rule constants).

Modules of rules are a means of grouping sets of reactions that repeatedly occur together within a model, and by moving modules into libraries they can be shared between sets of models. We use modules as a means of constraining model structure optimisation to biological plausible reaction interaction networks and maintaining a consistent level of detail across models.

### 1.4.2 Simulation

Simulation recreates the dynamics of a system as described by a model. Quantitative simulations enables measurements of model features changing in time which can be compared with observations of the real system for validation and predictive purposes. The Infobiotics Workbench simulator, MCSS, offers a choice of two types of quantitative simulations: deterministic numerical approximation with standard solvers, and execution of the model with stochastic simulation algorithms. In addition to providing a baseline implementation of the canonical Gillespie Direct Method, MCSS implements an optimised multi-compartmental SSA with queue [113] that takes advantage

of the compartmentalised nature of LPP system models by storing the next reaction to fire for each compartment in the heap and only recalculating the propensities of the reactions in the compartments where a reaction occurs, both compartments involved in a species translocation. This greatly improves performance, decreasing the simulation time of models with tens of thousands of compartments and hundreds of reactions and species per compartment.

In order to perform deterministic simulations MCSS derives a set of ordinary differential equations from the stochastic rules of the entire LPP system: each pool of identical objects in different compartments is treated as a separate continuous variable whose rate of change is determined by mass-action kinetics involving only the variables corresponding to reactants and products of those rules affecting the pool. A solution of the resultant equations is obtained using algorithms provided by the GNU Scientific Library (GSL) [48], including explicit 4th order Runge–Kutta and implicit ODE solvers.

When a model is simulated via the GUI, the output data file of a completed simulation is auto-loaded into the simulation results interface under a new tab, as shown in Fig. 1.5. The purpose of this interface is to enable the user to select a subset of the datapoints logged during a simulation (for some or all of the runs, species, compartments and timepoints), which can then be visualised using the provided time series, histogram or surface plotting functions (explained in detail below), or exported in various data formats for manipulation by third party software.

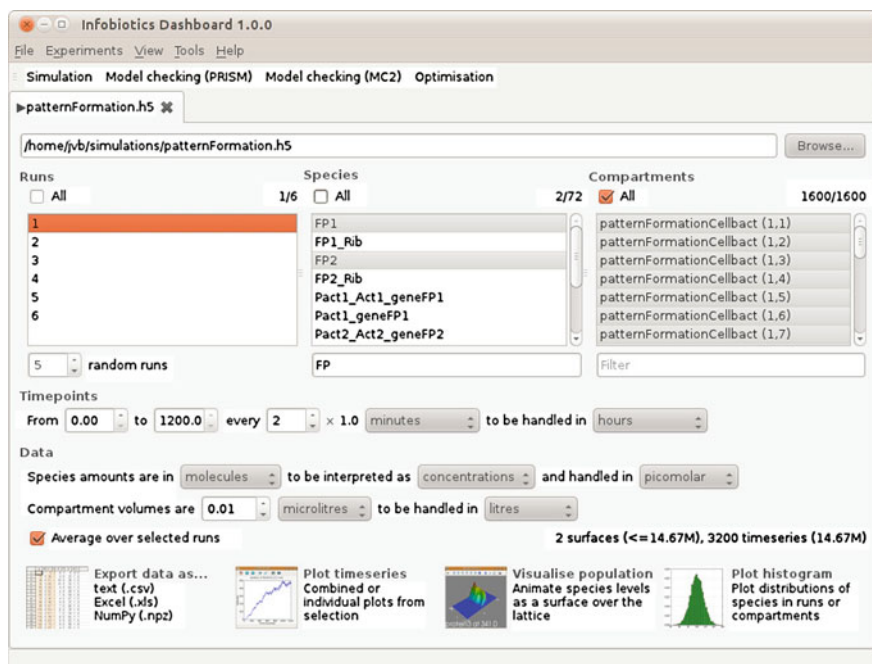


Fig. 1.5 The simulation results interface

The simulation GUI has the following useful features to make the analysis simple, customisable and reproducible:

- Individual entries, multiple or all runs, species and compartments can be selected.
- The list of species names can be sorted in either ascending or descending alphabetical order, and filtered by name.
- The list of compartments can similarly be sorted or filtered by name, and compartments can additionally be sorted by their X and Y positions on the lattice.
- The number of time points to use can be adjusted by changing the interval values of the `from`, `to` and `every` spinboxes.
- The *data units* of the model components can be set, and the *display units*, in which simulation results are to be handled and presented in plots, can be specified for timepoints, species quantities and compartment volumes. For instance, species amounts may be interpreted as either molecules, moles or concentrations, the choice determines which display units are available.
- The user can choose whether or not to average the amounts of each species in each compartment over the set of selected runs (default for stochastic simulations, hidden along with the list of runs for deterministic simulations). Averaging over many runs can approximate the deterministic outcome for systems where stochasticity is of lesser importance.
- The Dashboard displays the number of time series and surfaces, and estimate the memory requirements of each action, allowing the user to determine how quickly the action can be performed and whether the results will be comprehensible.
- The selected and rescaled datapoints can be exported from the Infobiotics Dashboard by clicking the `Export data as...` button to open a save file dialog limited to files with the extensions `.csv` (comma-separated value), `.xls` (Microsoft Excel) and `.npz` (NumPy).
- Distributions of the average quantity of each selected species at a single timepoint can be plotted as histograms for either each selected compartment over all selected runs, or each selected run over all selected compartments.
- With the time series plotting functionality, users can make exact (`combined`) or relative (`stacked/tiled`) quantitative comparisons of the temporal behaviour of multiple molecular species in multiple compartments, between several, or averaged over many, simulation runs. These plots can be exported as images for further comparison with experimental observations. Figure 1.6 shows the time series plotting interface for the *stacked* style. When working on a stacked or tiled plot, the `Refine time series selection` button will open a dialog in which the order and visibility of subplots can be adjusted.

When averaging over multiple runs, each line is the *sample mean* and each marker is overlaid with error bars of either the standard deviation of the sample (SD) or the confidence interval (CI) describing the accuracy of the standard deviation.

The figure toolbar provided by Matplotlib [84] enables zooming, panning, *Subplot configuration*: adjustment of the spacing between multiple plots and the figure



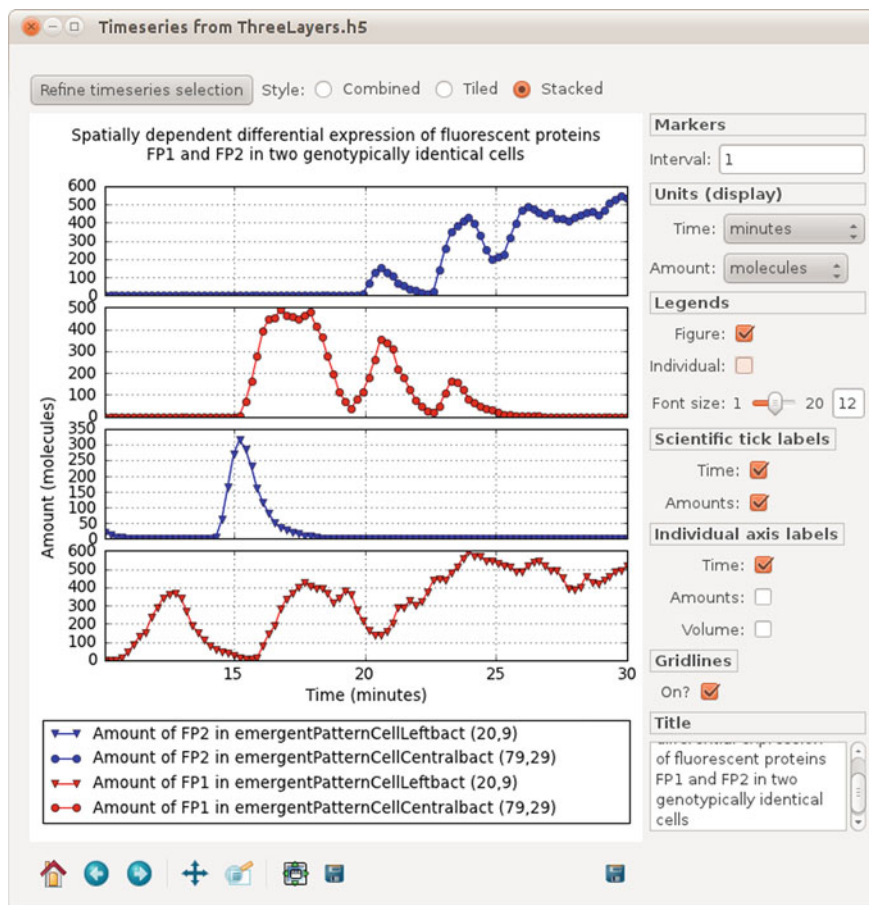
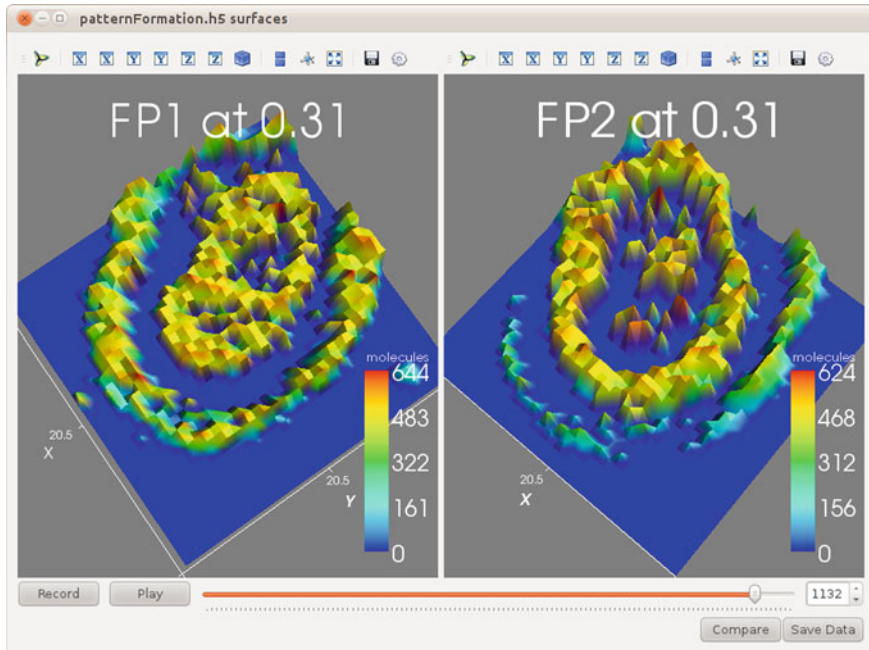


Fig. 1.6 Time series: stacked plot style

boundary and exporting plot image, as it appears for publication in bitmap and vector formats.

- The Infobiotics Dashboard enables users to visualise how species quantities change in time and 2D space by using 3D heat-mapped meshes or *surface* (where the vertices of the mesh correspond to model lattice points and the height of the peaks to the species quantities), to capture the distribution of each selected species over the model at a single timepoint. Multiple surfaces, one per species, each corresponding to particular species, can be visualized simultaneously side-by-side for qualitative comparison. The overlaid scalar bars map heat as colour to quantities.

Figure 1.7 shows an example in which two surfaces plots of 1,600 compartments (40 x 40) are rendered. Time is progressed either manually, by dragging the time-point index slider, or automatically using the Play/Pause button.



**Fig. 1.7** Surface plots showing expression patterns of two fluorescent proteins

Surfaces plots provide an intuitive means of qualitatively gauging the behaviour of population level models, that may (cautiously) be compared to microscopy data.

### 1.4.3 Model Checking

By encoding a biological system into a formal system we can make inferences about the system and discover novel knowledge about the system properties. A central mission of executable biology is to apply *model checking* techniques to biological systems. Model checking goes beyond repeated simulation and observation to provide a formal verification method that the model of real-life system is correct in all circumstances. Namely, model checking a system means exhaustively enumerating all of its possible states over the range of possible inputs and transitions to produce every possible sequence of events, which cannot be done using simulation.

Probabilistic model checking is a probabilistic variant of classical model checking augmented with quantitative information regarding the likelihood that certain transitions occur and the times which they do so. Probabilistic model checking works with *Discrete time Markov Chains (DTMCs)*, *Continuous time Markov chains (CTMCs)* or *Markov Decision Processes (MDPs)*. A continuous time Markov chain (CTMC) is



defined by a set of *states*, a set of *initial* states and a *transition rate matrix* from which the rate at which a transition occurs between each pair of states is taken as a parameter of an exponential distribution. Queries which check model properties are defined as logical statements, often *probabilistic logics*: CSL (*Continuous Stochastic Logic*) [3] for CTMCs, PCTL (*Probabilistic Computation Tree Logic*) [63] for DTMCs and MDPs.

The Infobiotics Workbench is equipped with a model checking module, called PMODELCHECKER. Properties of stochastic P system models can be expressed as probabilistic logic formulas and automatically verified using third party model checking softwares, namely PRISM [74, 76] and MC2 [33–35]. PMODELCHECKER [110] extends this capability to LPP system models by acting as wrapper interface between LPP systems and the model checkers PRISM and MC2.

To perform probabilistic model checking with PRISM, LPP systems are loaded and automatically converted into a Reactive Modules specification (a CTMC) [2] that PRISM can accept as input. Parameters are created for the lower and upper bounds of the number of molecules of each species in each compartment: the user defined values of which are used to constrain the potential state space of the PRISM model. PRISM is then called to perform *statistical* model checking using its own discrete event simulator, performing simulations up to a specified maximum number of runs or a confidence threshold (typically 95 %). The state space and the generated transitions matrix can also be used to “Build” an efficient representation of the complete Markov chain and then “Verify” whether each property is satisfied in all states of the model. Such exhaustive verification is generally infeasible for all but very small models due to the size of the underlying CTMC, but can be useful for checking critical components of small reaction networks, such as synthetic bioparts.

To perform *statistical* model checking with MC2, previous simulation results can be reused or a new simulation can be performed with a large number of runs to achieve higher confidence in the model checking results. With model checking, properties such as the probability of a species exceeding a certain threshold after a certain time can be determined to a specified degree of confidence (corresponding to the number of independent simulation runs for simulative model checking).

The Infobiotics Dashboard provides two parameterisation interfaces to PMODELCHECKER, one for each of the model checkers it uses, as some of the parameters are specific to one but not the other. Figure 1.8 illustrates the PRISM interface showing the P system model, Temporal Formulas and Results file parameter widgets.

Multiple formulas can be loaded from, and must be saved to, a file. The currently selected formula can be edited or removed, or a new formula added via the respective buttons. Formulas are edited manually and can be parameterised with variables that are finite ranges with equal steps.

Once a model checking experiment has completed the results interface is loaded from the file specified by the `results_file` parameter. The output is the same for either model checking experiment: for each formula a list of the probability of each property being fulfilled for each combination of formula parameters, usually time plus several others (e.g. Fig. 1.9). The varying probabilities of each property can be plotted in two ways: a 2D plot of the probability that the property is satisfied against

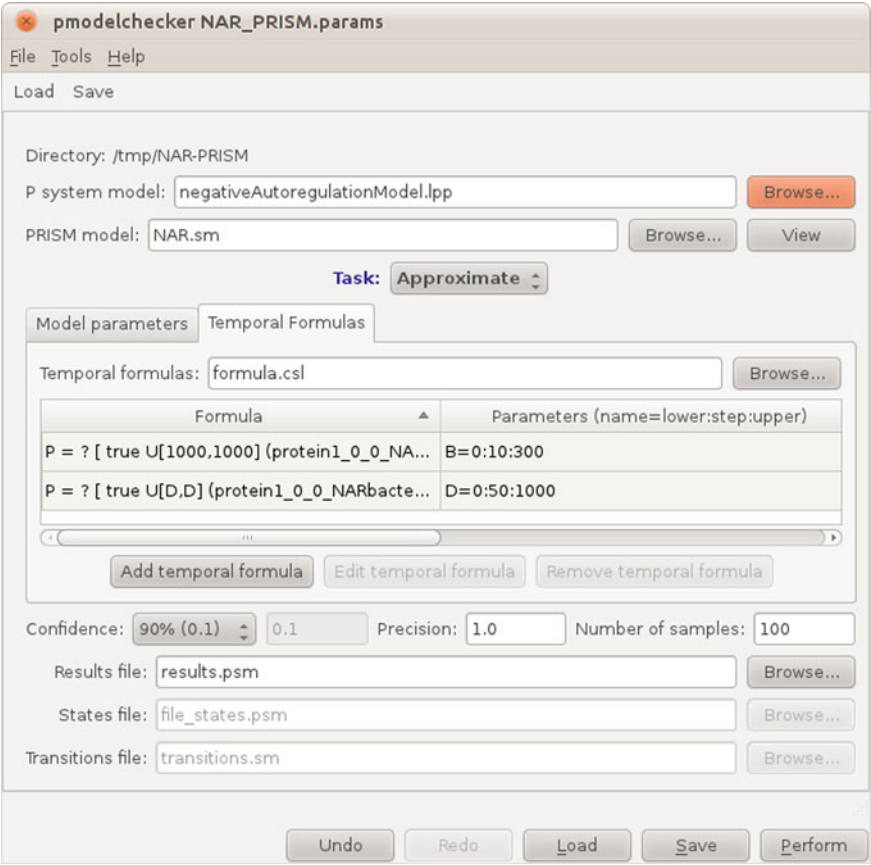


Fig. 1.8 PMODELCHICKER parameterisation interfaces

all values of one variable (Fig. 1.9a) or a 3D plot of probability against all values two variables (Fig. 1.9b), at a single value of each remaining variable. The constant values of the remaining variables can be set using sliders which are dynamically added to the results interface above the plot depending on availability and the currently selected axis variables. In this way both 2D and 3D plots can be used to visualise queries with greater numbers of variables, enabling the results of N-dimensional queries to be interrogated in a consistent manner.

1.4.4 Optimisation

Both stochastic and deterministic models are dependent on the correct model structure and accurate rate constants to accurately reproduce cellular behaviour. Unfortunately well-characterised rate constants are in very short supply, and those that are

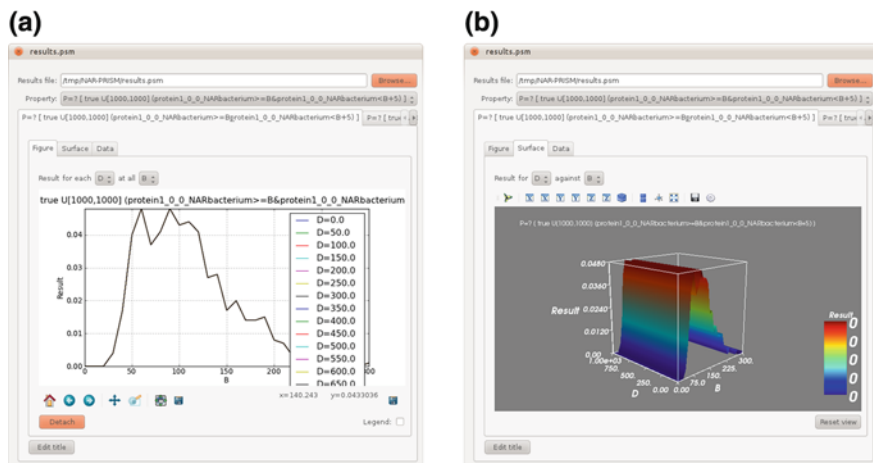


Fig. 1.9 Model checking experiments results interface. **a** 1 variable 2D plot. **b** 2 variable 3D plot

known for some models are used as ersatz values in models of similar systems. In the scenario, where the components and interactions are known but other parameters are not, it is acceptable to try estimate the rate constants using parameter optimisation to fit model dynamics to laboratory observations.

OPTIMIZER is the model optimisation component of the Infobiotics Workbench. Optimisation is the process of maximising or minimising certain criteria by adjusting variable components of a model, fitting simulated behaviour (quantitative measurements sampled at various time intervals) to observed or desired behaviour in the case of natural or synthetic biological systems respectively. There are two aspects of P system models that can be readily varied to optimise temporal behaviour:

1. numerical model parameters—the values of the stochastic rate constants associated with rules can be tuned to fit the given target,
2. model structure—the composition of the rulesets governing the possible state transitions of the compartments can be altered to produce alternative reaction networks that recreate the target dynamics more precisely.

Both seek to *minimise* the distance between the stochastically simulated quantities of molecular species and a set of user-provided values of the same species at each target timepoint; a quantitative means of evaluating the fitness of candidate models and discriminating between them in a automated manner.

OPTIMIZER searches the parameter and structure spaces of *single compartment* stochastic P systems with implementations of state-of-the-art population-based optimisation algorithms: Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [62], Estimation of Distribution Algorithms (EDA), Differential Evolution (DE) [119] and Genetic Algorithms (GA) [59]. Optimisation is limited to single compartment models, partly due to the increased complexity of algorithmically manipulating spatially distributed or hierarchically organised compartmental

structures (and the distinction made between these by the LPP formalism), but more pragmatically because repeated stochastic simulation of each individual in a population of (potentially unfeasible) single compartment models (with suboptimal rate constants) is very computationally expensive. Simulating many copies of those compartments, interacting on a 2D lattice would multiply the cost and providing suitable or accurate target data would be difficult also. Thus model optimisation is generally only tractable with smaller models (as with model verification). However, submodels can be optimised in isolation and then reintegrated, provided they can be decoupled: the assumption made by the modularised, engineering approach to synthetic biology.

OPTIMIZER uses a *nested genetic algorithm* [20, 111] to generate a set of candidate models, initially by random choice and thereafter by mutating the fittest individuals of the previous generation, performing several rounds of parameter optimisation on each individual to ensure that the candidates are given a fair chance of fitting the desired behaviour (as previous rate constants may be unsuited to the updated reaction network) before using the final fitness to select the next generation.

The output of an optimisation experiment is the fittest model produced. For a visual comparison of the output models suitability and the optimisation algorithms success, time series of the target and the optimised output are plotted for each species, as shown in Fig. 1.10. A summary of the experiments inputs and the modules that comprise the optimised model are captured from OPTIMIZER and displayed alongside the time series.

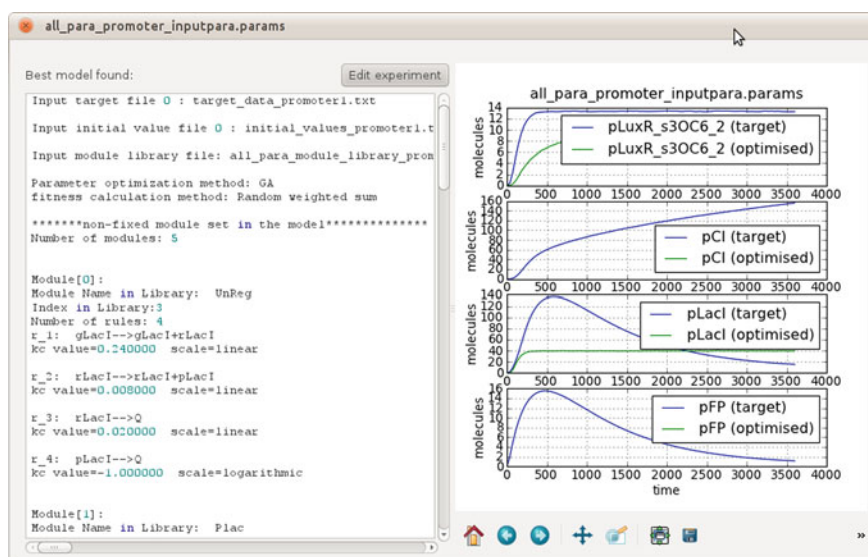


Fig. 1.10 OPTIMIZER results interface

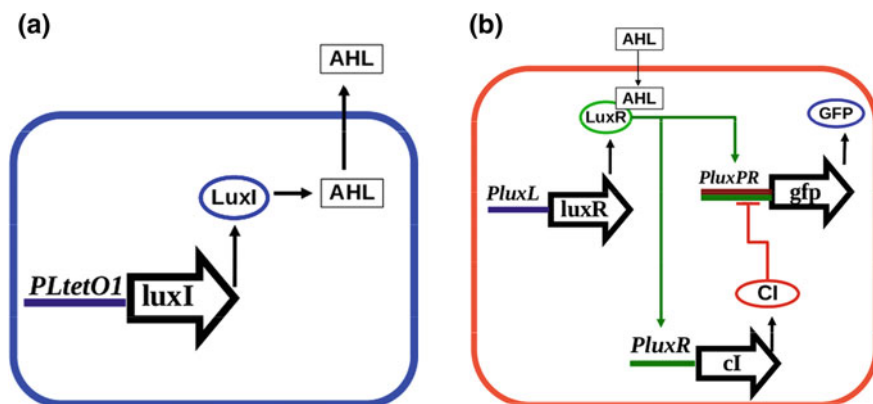
## 1.5 Case Study

In this section, we demonstrate the use of the IBW features in a case study. Here, we select the *pulse generator* example, which consists of the synthetic bacterial colony designed by Ron Weiss' group in Refs. [4, 5]. This model implements the propagation of a wave of gene expression in a bacterial colony. For other applications of this modelling see Refs. [45, 121].

The pulse generator consists of two different bacterial strains, *sender cells* and *pulsing cells* (see Fig. 1.11):

- Sender cells contain the gene `luxI` from *Vibrio fischeri*. This gene codifies the enzyme LuxI responsible for the synthesis of the molecular signal 3OC6-HSL (AHL). The `luxI` gene is expressed constitutively under the regulation of the promoter `PLtetO1` from the tetracycline resistance transposon.
- Pulsing cells contain the `luxR` gene from *Vibrio fischeri* that codifies the 3OC6-HSL receptor protein LuxR. This gene is under the constitutive expression of the promoter `PluxL`. It also contains the gene `ci` from lambda phage codifying the repressor CI under the regulation of the promoter `PluxR` that is activated upon binding of the transcription factor `LuxR_3OC6_2`. Finally, this bacterial strain carries the gene `gfp` that codifies the green fluorescent protein under the regulation of the synthetic promoter `PluxPR` combining the `Plux` promoter (activated by the transcription factor `LuxR_3OC6_2`) and the `PR` promoter from lambda phage (repressed by the transcription factor `CI`).

The bacterial strains above are distributed in a specific spatial distribution. As shown in Fig. 1.12, sender cells are located at one end of the bacterial colony and the rest of the system is filled with pulsing cells.



**Fig. 1.11** Two different bacterial strains of the pulse generator. **a** Sender cell **b** Pulsing cell

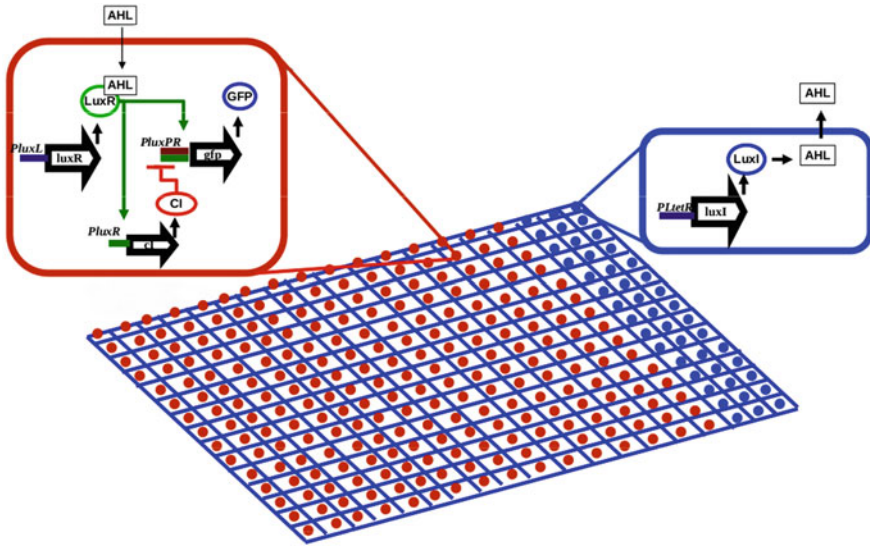


Fig. 1.12 Spatial distribution of sender and pulsing cells

### 1.5.1 LPP Model

As discussed in Sect. 1.3, the Infobiotics Workbench accepts system models in LPP language to activate its features. LPP systems are an extension of stochastic P systems with spacial dimension. Namely, they allow us to model a 2D geometric *lattice* on which a population of stochastic P systems could be placed and over which molecules could be *translocated*.

Here, we give a short account on the LPP model. Our model of the pulse generator uses a module library describing the regulation of the different promoters used in the two bacterial strains. An additional module library describing several post-transcriptional regulatory mechanisms is also used in our model. The bacterial strain, sender cell, producing the signal 3OC6-HSL (AHL) is modelled using the SP-system model. The bacterial strain, pulsing cell, producing a pulse of GFP protein as a response to the signal 3OC6-HSL (AHL) is modelled using another SP-system model. In order to prevent any modelling issues in our framework, we add an extra cell to represent the *boundary* of the system. The geometry of a bacterial colony of the cell type or bacterial strain represented in the previous model is captured using a rectangular lattice. Finally, the model of the synthetical bacterial colony is obtained by distributing cellular clones of the sender cell strain at one end of the lattice and cellular clones of the pulsing cell strain over the rest of the points.

## 1.5.2 Simulations

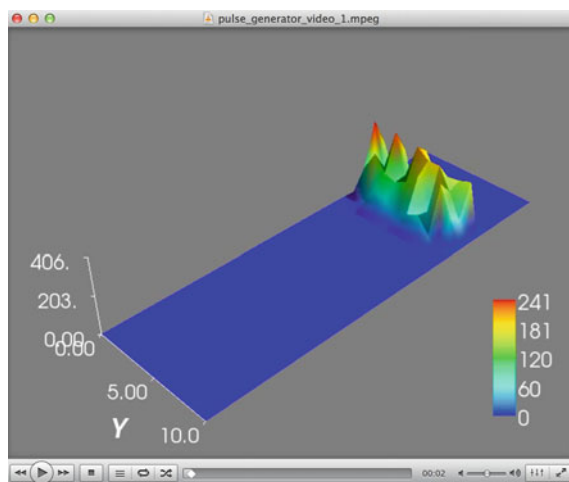
The final model has 341 compartments ( $11 \times 31$ ), 28 molecular species and 8,783 rules in total. 5 stochastic simulation runs of 800 simulated seconds required an average of 2 min and 4 s wall clock time on a single 2.20 GHz core of an Intel(R) Core(TM) i7-2670QM. The enhanced multi-compartmental stochastic simulation algorithm performed 65,679,239 total reactions per run on average, achieving a rate of 528,968 reactions per second. It should be noted the time required to simulate a model is highly dependent on the structure of the reaction network in addition to the number of the compartments and reactions, and that fluctuations in the number of molecules in the system as its state changes can dramatically impact the rate of simulation.

The IBW interface enables the user to select a subset of the datapoints logged during a simulation for each species, which can then be visualised using the provided time series, histogram or surface plotting functions.

Figures 1.13 and 1.14 show the spatial propagation of a pulse of GFP over the bacterial colony and a single pulsing cell, respectively. As the figures show, the GFP protein propagates through pulsing cells until the concentration level drops to 0.

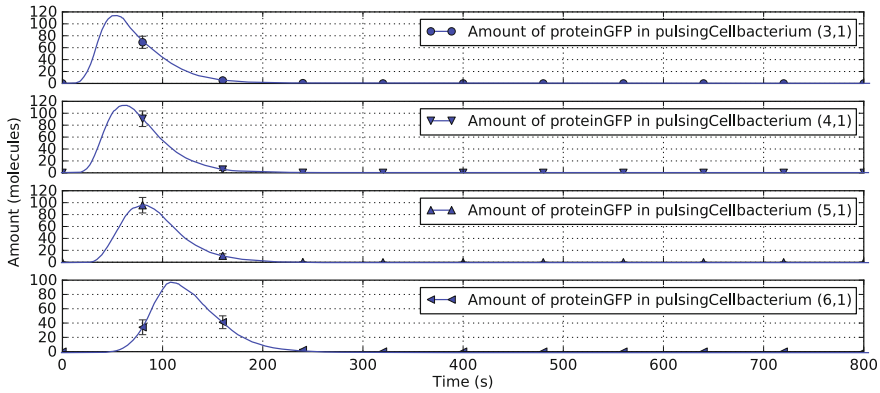
Figure 1.15 shows the concentration of the `PluxPR_LuxR2_GFP` promoter, which regulates the expression of the protein GFP, in different cells. As shown in the figure, the concentration first increases, and then permanently becomes zero after 100 s. This explains the behaviour observed in Fig 1.14, because when the promoter concentration becomes zero, the protein GFP cannot be expressed.

Figure 1.16 shows the signal molecule `signal30C6` amount over time. The figure suggests that the further away the pulsing cells are from the sender cells the less likely they are to produce a pulse.

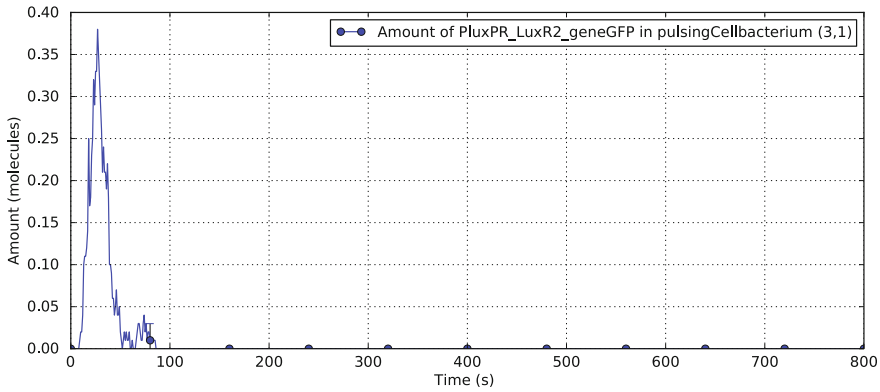


**Fig. 1.13** Spatial propagation of GFP over the bacterial colony





**Fig. 1.14** Propagation of GFP over a pulsing cell



**Fig. 1.15** signal3OC6 level over time

### 1.5.3 Model Checking

We now present our results of the system analysis using the probabilistic model checking techniques. Before presenting the experiments performed, we give a brief overview on the property specification in PRISM and MC2 model checkers.

#### PRISM

In PRISM, properties are specified in Continuous Stochastic Logic (CSL) [3]—an extension of Probabilistic Continuous Time Logic (PCTL) [63] for CTMCs.

CSL formulas are interpreted over CTMCs. The execution of a CTMC constructs a set of *paths*, which are *infinite* sequences of states. Apart from the usual operators from classical logic such as  $\wedge$  (and),  $\vee$  (or) and  $\Rightarrow$  (implies), CSL has the *probabilistic*



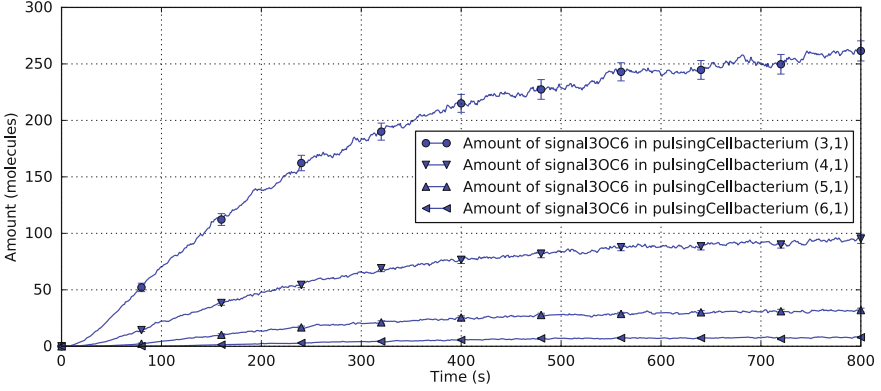


Fig. 1.16 signal3OC6 level over time

operator  $\mathbf{P}_{\sim r}$ , where  $0 \leq r \leq 1$  is a *probability bound* and  $\sim \in \{<, >, \leq, \geq, =\}$ .<sup>2</sup> Intuitively, a state,  $s$ , of a model satisfies  $\mathbf{P}_{\sim r}[\varphi]$  if, and only if, the probability of taking a path from  $s$  satisfying the *path formula*  $\varphi$  is bounded by ' $\sim r$ '. The following path formulas  $\varphi$  are allowed:  $\mathbf{X}\phi$ ;  $\mathbf{F}\phi$ ;  $\mathbf{G}\phi$ ;  $\phi\mathbf{U}\psi$ ; and  $\phi\mathbf{U}^{\leq k}\psi$  (Note that the operators  $\mathbf{F}\phi$  and  $\mathbf{G}\phi$  can actually be derived from  $\phi\mathbf{U}\psi$ ).

As an example, the property that “the probability of  $\varphi$  eventually occurring is greater than or equal to  $b$ ” can be expressed in CSL as follows:

$$\mathbf{P}_{\geq b}[\text{true } \mathbf{U} \varphi].$$

The informal meanings of such formulas are:

- $\mathbf{X}\phi$  is true at a state on a path if, and only if,  $\phi$  is satisfied in the next state on the path;
- $\mathbf{F}\phi$  is true at a state on a path if, and only if,  $\phi$  holds at some present/future state on that path;
- $\mathbf{G}\phi$  is true at a state on a path if, and only if,  $\phi$  holds at all present/future states on that path;
- $\phi\mathbf{U}\psi$  is true at a state on a path if, and only if,  $\phi$  holds on the path up until  $\psi$  holds; and
- $\phi\mathbf{U}^{\leq k}\psi$  is true at a state on a path if, and only if,  $\psi$  satisfied within  $k$  steps on the path and  $\phi$  is true up until that moment.

As well as the probabilistic operator  $\mathbf{P}_{\sim r}$ , CSL also includes  $\mathbf{S}_{\sim r}$  and  $\mathbf{R}_{\sim r}$  operators to express properties regarding the *steady-state* behaviour and expected values of *rewards* respectively. There are four different types of reward formulas, which are the *reachability* reward  $\mathbf{R}_{\sim r}[\mathbf{F}\varphi]$ , *cumulative* reward  $\mathbf{R}_{\sim r}[\mathbf{C} \leq t]$ , *instantaneous* reward  $\mathbf{R}_{\sim r}[\mathbf{I} = t]$  and *steady-state* reward  $\mathbf{R}_{\sim r}[\mathbf{S}]$ . The informal semantics of these formulas are given below [75]:

<sup>2</sup> The  $\mathbf{P}_{\sim r}$  operator is the probabilistic counter-part of path-quantifiers  $\forall$  and  $\exists$  of CTL.

- $\mathbf{S}_{\sim r}[\varphi]$  asserts that the steady-state probability of being in a state satisfying  $\varphi$  meets the bound  $\sim r$ .
- $\mathbf{R}_{\sim r}[\mathbf{F}\varphi]$  the expected reward accumulated before a state satisfying  $\varphi$  is reached meets the bound  $\sim r$ .
- $\mathbf{R}_{\sim r}[\mathbf{C} \leq t]$  refers to the expected reward accumulated up until time  $t$ .
- $\mathbf{R}_{\sim r}[\mathbf{I} = t]$  asserts that the expected value of the state reward at time instant  $t$  meets the bound  $\sim r$ .
- $\mathbf{R}_{\sim r}[\mathbf{S}]$  asserts the long-run average expected reward meets the bound  $\sim r$ .

## MC2

In MC2, properties are specified in a variant of Probabilistic Linear Temporal Logic (PLTL) [90] (which is a probabilistic extension of Linear Temporal Logic (LTL) [100]). This variant is called PLTL<sub>c</sub> [35], the discrete time steps of which correspond to the logging interval of the simulation.

PLTL<sub>c</sub> formulas are interpreted over a finite set of finite paths (e.g., simulation traces and time series). The PLTL<sub>c</sub> language extends the syntax of LTL with *numerical constraints* and a *probability operator*. Therefore, in addition to the standard boolean operators (e.g.,  $\wedge$ ,  $\vee$  and  $\Rightarrow$ ) and temporal operators (e.g.,  $\mathbf{X}\phi$ ,  $\mathbf{F}\phi$ ,  $\mathbf{G}\phi$  and  $\phi\mathbf{U}\psi$ ), PLTL<sub>c</sub> includes numerical constraints in the form of  $value \sim value$  ( $\sim \in \{<, >, \leq, \geq, =, \neq\}$ ), where *value* is defined as follows [35]:

```
value ::=
  Int | Real | [molecule] | max[molecule] | d[molecule] | $fVariable
  value + value | value - value | value * value | value/value
```

where *Int* denotes integer numbers, *Real* denotes real numbers, *\$fVariable* denotes *free* variables, *[molecule]* denotes molecular concentrations of biochemical species, *max[molecule]* denotes a function which “operates over all the values of a species and returns the maximum of the species value in simulation runs” [35] and *d[molecule]* denotes a function which returns “the derivative of the concentration of the species at each time point” [35] in a simulation run.

PLTL<sub>c</sub> also includes the *probabilistic operators*  $\mathbf{P}_{\sim r}$ , where  $0 \leq r \leq 1$  is a *probability bound* and  $\sim \in \{<, >, \leq, \geq\}$  (without equality “=”).

## Property Patterns

Model checking is a very useful method to analyse the expected behaviour of biological models. It formalises simulation and observation to verify that the model of a biological system is correct in all circumstances.

Although model checking is a well-established and widely used formal method, it requires formulating properties in a dedicated formal syntax, and hence, formal specification can be a very complex and error-prone task especially for non-expert

users. For example, the question *what is the probability that the number of molecules exceeds 100 within 60 min in 90% of the cases* is expressed in CSL as follows:

$$P_{=0.9}[true \ U^{\leq 60} \ molecules \geq 100].$$

Clearly, this property is very simple to express in natural language, but it is difficult (for non-experts) to specify formally as it requires familiarity with the syntax of the formalism. In the case of more complex properties, the formal specification of certain properties might become a more cumbersome task.

To facilitate property specification and therefore to increase the accessibility of useful capabilities of model checking to a wider group of users, we have developed the NLQ (*Natural Language Query*) tool, which converts *natural language queries* into their corresponding formal specification language. NLQ is based on the prototype tool introduced in Ref. [36] with extra added features and support for probabilistic logics used in the model checking module of the Infobiotics Dashboard. Using the NLQ tool, users can create a set of properties simply by manipulating a configurable form with graphical user interface elements such as drop-down lists and text fields.<sup>3</sup>

Another important feature of the NLQ tool is that it provides users with a set of so called *property patterns* based on most frequent properties in the model checking study. Since the seminal paper of Dwyer et al. [37], there have been many developments in categorising recurring properties into specific property patterns, which can be considered as generic representations of instances of numerous properties utilised in different contexts. Indeed, [37] surveyed more than 500 temporal properties and categorised them into a handful of property patterns. [73] extended this work to include real-time specification patterns. Reference [60] presented a similar pattern system for probabilistic properties.

Some of the patterns used in the NLQ tool is shown in Table 1.1. These patterns provide a coherent set of templates, which guide users to construct formal expressions to represent desired properties.

## Experiments

We now present the results of the probabilistic model checking experiments we carried out. Due to the well-known scalability issues that model checkers suffer we reduced the size of the lattice to  $4 \times 8$ , where the surrounding cells are boundary cells and  $2 \times 2$ -sender cells are located inside at one edge, which are followed by  $4 \times 2$ -pulsing cells (see Fig. 1.12).

Table 1.2 shows the informal specifications of the properties and the corresponding CSL formulas that PRISM accepts as input. It presents query results for each of Prop. 1, 2, 3 and 4. The verification results of Prop. 5, 6, 7 and 8 are illustrated as a 2D plot in Fig. 1.17, where *Row n* denotes the *n*th row of the pulsing cells in the lattice, *T*

<sup>3</sup> At the moment, the NLQ tool is not integrated into the Infobiotics Workbench. But, the properties it generates can be directly used in IBW's model checking component.

**Table 1.1** Property patterns

Pattern	Example
<i>Occurrence</i>	The number of molecules of $x$ exceeds 100 within 50 s in 90 % of the cases
<i>Until</i>	Until the concentration of the promoter $x$ is greater than 0.5, the probability of expressing the gene $x$ is less than 0.01
<i>Universality</i>	The concentration of the signalling protein never drops below the threshold
<i>Response</i>	If the concentration of the repressor protein is more than 0.5, then the probability that the regulation of the protein will be repressed is greater than 0.9
<i>Precedence</i>	Only, after the concentration of the repressor protein is more than 0.5, the probability that the regulation of the protein will be repressed is greater than 0.9
<i>Steady State</i>	In the steady state, the probability that the concentration of the signalling protein is more than 1nM is greater than 0.9
<i>Reward</i>	The expected concentration of the signalling protein at the time instant 100 is between 0.9 and 1.0 nM

denotes time and y-axis represents the verification result of the corresponding PRISM query.

Based on these results, we have made some observations. Firstly, as Fig. 1.17a, c suggest, the GFP protein propagates through the pulsing cells. Namely, the GFP protein is first observed in the rows closer to the sender cells, then the concentration level drops until it permanently becomes zero. On the other hand, the concentration level in the next rows shows a similar pattern with some delay, which is proportional to the distance of the row to the sender cells. This behaviour can also be observed from Prop. 1, 2, 3 and 4. Figure 1.17d also suggests that the further away the pulsing cells are from the sender cells the less likely they are of producing a pulse. Clearly, these results are in line with the simulation results discussed previously.

As verification experiments result show, model checking can provide more insight into system models than simulations to analyse system dynamics and complex behaviour by means of formal queries. Table 1.2 illustrates how the NLQ tool automatically translates informal queries into formal representations, which can be directly used to query model checkers.

### 1.5.4 Supplementary Material

The complete model and experimental results of the pulse generator example can be downloaded from the IBW website [105]. These include LPP model files, simulation parameters, simulation results, PRISM model file, model checking parameters, a list of PRISM properties and model checking experimental results. The interested readers can try running the experiments themselves. The “README.txt” file provides a detailed guidance on how to perform the same or similar experiments.

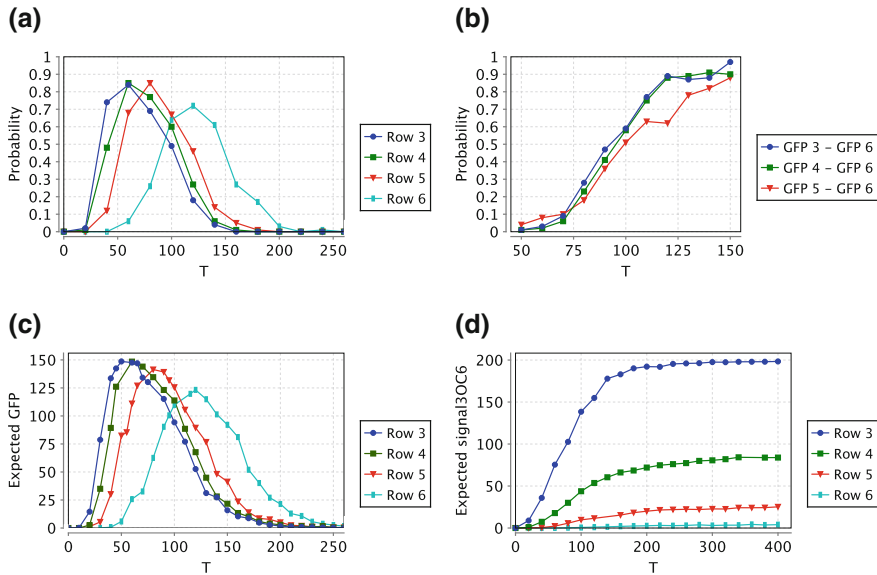
**Table 1.2** PRISM properties

Prop.	Informal and formal specification	PRISM vrf.
1	Probability that GFP concentration at row 3 exceeds 100 within 50 s. $\mathbf{P}_{=?}[true \ U^{\leq 50} \text{GFP\_pulsing\_3} \geq 100]$	0.87
2	Probability that GFP concentration at row 3 exceeds 100 between 50 and 100 s. $\mathbf{P}_{=?}[true \ U^{[50,100]} \text{GFP\_pulsing\_3} \geq 100]$	0.92
3	Probability that GFP protein at row 3 always exists after 200 s. $\mathbf{P}_{=?}[\mathbf{G}^{\geq 200}(\text{GFP\_pulsing\_3} > 0)]$	0.0
4	Probability that GFP concentration at row 5 stays greater than 100 before GFP concentration at row 3 exceeds 100. $\mathbf{P}_{=?}[\text{GFP\_pulsing\_5} \geq 100 \ \mathbf{W} \ \text{GFP\_pulsing\_3} \geq 100]$	0.0
5	Probability that GFP concentration at row $n \in \{3, 4, 5, 6\}$ exceeds 100 at instant $T$ . $\mathbf{P}_{=?}[true \ U^{[T,T]} \text{GFP\_pulsing\_n} \geq 100]$	see Fig. 1.17a
6	Probability that GFP concentration at row $n \in \{3, 4, 5\}$ stays greater than GFP concentration at row 6 until time instant is $T$ where GFP concentration at row 6 exceeds GFP concentration at row $n$ . $\mathbf{P}_{=?}[\text{GFP\_pulsing\_n} \geq \text{GFP\_pulsing\_6} \ U^{[T,T]} \text{GFP\_pulsing\_6} > \text{GFP\_pulsing\_n}]$	see Fig. 1.17b
7	Expected GFP concentration at row $n \in \{3, 4, 5, 6\}$ at instant $T$ . $\mathbf{R}\{“\text{GFP\_pulsing\_n}”\}_{=?}[\mathbf{I} = T]$	see Fig. 1.17c
8	Expected signal3OC6 concentration at row $n \in \{3, 4, 5, 6\}$ at instant $T$ . $\mathbf{R}\{“\text{signal3OC6\_pulsing\_n}”\}_{=?}[\mathbf{I} = T]$	see Fig. 1.17d

## 1.6 Discussions and Conclusions

In this last section we compare the best known tools based on the P system modelling paradigm which are used in system and synthetic biology. In the last part further developments for IBW are presented.

As we have seen so far, IBW is a complex software environment combining the power and flexibility of a formal modelling framework based on stochastic P systems enhanced with a lattice-based geometry and a modular way of grouping rules. It also includes an advanced formal verification component consisting of some probabilistic and stochastic model checking tools, PRISM and MC2, together with a natural language pattern facility allowing to formulate various queries in a free style without



**Fig. 1.17** Model checking experiment results. **a** Probability of GFP exceeds threshold (Prop. 5). **b** Probability of relative GFP (Prop. 6). **c** Expected GFP protein (Prop. 7). **d** Expected signal3OC6 (Prop. 8)

paying attention to specific syntactic constraints. The other key component of this tool is a model structure and parameter optimisation engine. These three components are fully integrated into an environment where they smoothly communicate, models can be edited and results of various experiments are visualised according to a broad range of options.

In what follows we compare the IBW set of functions with other similar P systems based modelling and analysis software platforms presented in Sect. 1.2.3. In order to assess the modelling capabilities of these tools with respect to their flexibility, analysis power and efficiency, we have considered features like modularisation, formal verification capability, structure and/or parameter optimisation aspects and the option to execute the simulation on parallel hardware architectures. All the considered tools benefit from an integrated development environment (IDE) with different levels of complexity. The results of the assessment are presented in Table 1.3.

It is difficult to compare the expressiveness of the modelling languages used by these tools, as although they all use the same P systems paradigm, they implement different features - some use deterministic execution style [11], whereas many rely on probabilistic or stochastic behaviour [7, 14, 23, 28, 67]; SRSim uses strings as opposed to all the others employing multisets; some use explicitly geometric elements [14, 67] or a topology of the environment [28], but the others make use only of the membrane structure.

**Table 1.3** Tools comparison

Tool	IDE	Modules	Verification	Optimisation	Parallel
MetaPlab	Yes	No	No	Yes	No
MeCoSim	Yes	?	No	No	Yes
BioSimWare	Yes	No	No	Yes	Yes
Cyto-Sim	Yes	No	Yes	No	No
SRSim	Yes	Yes	No	No	No
IBW	Yes	Yes	Yes	Yes	Yes

It is well-known that most of the systems and synthetic biology models are complex, with a rich combinatorics of biochemical interactions and certain motifs occurring. A way of coping with this aspect is to provide some modularisation capabilities. P systems by their very nature introduce a type of modularisation by defining compartments. In many cases these are utilised as topological components rather than functional units and do not provide adequate mechanisms to instantiate units of functionality with the same behaviour, but with different biochemical elements or concentrations. IBW and SRSim make use of modules directly in their specification languages, MeCoSim through its associated P-Lingua language define them as blocks of rules expressing a certain behaviour, without an explicit instantiation mechanism.

Simulations represent the key component of all these tools and these are quite different as the simulation methods depend on the semantics associated to the P systems utilised by the tools. We can not compare them as, on the one hand, there is not much data published regarding the performance of the simulators, and the size of the models, and, on the other hand, the scope of them is quite broad and different.

The results of the simulations require a form of validation, through experiments, or in depth analysis, with mathematical and/or computational instruments, complementing the simulation. Such an analysis method is the formal verification approach based on computational models [42, 43], especially model checking. So far, only IBW and, very recently, Cyto-Sim [23] support this type of analysis. In IBW this analysis is fully integrated with the rest, the translation into PRISM is automatically obtained from the specification and the queries formulated for each model are expressed using natural language patterns. Another feature of these tools that helps post-simulation analysis is the visualisation capability. This can be observed in some of these tools, MeCoSim, SRSim, MetaPlab, IBW, as being fully integrated with the other components.

Biological systems in contrast to complex engineering systems are in many cases not fully specified. At least two aspects are not always known, the kinetic rates of some interactions and the structure of certain components. These issues are overcome by employing optimisation methods for approximating the unknown aspects. MetaPlab uses such methods to approximate functions associated to rules in MP models, BioSimWare deals with parameter estimation [12] and IBW provides mechanisms for parameter estimation and model structure optimisation in the case of

stochastic systems. Recently, it is reported the possibility of using similar methods for BioSimWare [10].

Complex simulations require better algorithms implementing various semantics associated to P systems models and also the use of novel technologies. In the last years there have been investigations related to the use of parallel hardware architectures for speeding-up simulations. IBW has a parallel version that distributes simulation runs over HPC clusters. BioSimWare has a version running on distributed architectures such as grid and CUDA [8]. MeCoSim/P-Lingua platform uses CUDA for PDP systems showing in certain cases significant increase in speed and new exciting research avenues [83]. However, this facility is not fully integrated in the software platform.

**Acknowledgments** JB, JT, SK, NK and MG acknowledge the support provided for synthetic biology research by EPSRC ROADBLOCK project (EP/I031642/1 & EP/I031812/1), EPSRC AUDACIOUS project (EP/J004111/1) and FP7 STREP CADMAD project. JR-C acknowledges support from Cellular Computing Applications into Systems and Synthetic Biology, TIN2009-13192, and Computational Modelling and Simulation in Systems Biology, P08-TIC-04200. MG was also partially supported by the MuVet project, (CNCS-UEFISCDI), grant number PN-II-ID-PCE-2011-3-0688. Some parts of this paper are based on the first author's Ph.D. thesis [13].

## References

1. U. Alon, Network motifs: theory and experimental approaches. *Nat. Rev. Genet.* **8**(6), 450–461 (2007)
2. R. Alur, T. Henzinger, Reactive modules. *Formal Methods Syst. Des.* **15**, 7–48 (1999)
3. C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Softw. Eng.* **29**, 524–541 (2003)
4. S. Basu, Y. Gerchman, C.H. Collins, F.H. Arnold, R. Weiss, A synthetic multicellular system for programmed pattern formation. *Nature* **434**, 1130–1134 (2005)
5. S. Basu, R. Mehreja, S. Thiberge, M.-T. Chen, R. Weiss, Spatiotemporal control of gene expression with pulse-generating networks. *PNAS* **101**(17), 6355–6360 (2004)
6. F. Bernardini, M. Gheorghe, F. Romero-Campero, N. Walkinshaw, A hybrid approach to modelling biological systems, in *Proceedings of 8th Workshop on Membrane Computing*, vol. 4860 (LNCS/Springer, 2007), pp. 138–159
7. D. Besozzi, P. Cazzaniga, G. Mauri, D. Pescini, BioSimWare : a software for the modeling, simulation and analysis of biological systems, in *CMC 2010*, vol. 6501 (LNCS/Springer, 2010), pp. 119–143
8. D. Besozzi, P. Cazzaniga, G. Mauri, D. Pescini, BioSimWare: a P systems-based simulation environment for biological systems, in *11th International Conference on Membrane Computing 2010*, vol. 6501 (LNCS, 2010), pp. 119–143
9. D. Besozzi, P. Cazzaniga, D. Pescini, G. Mauri, Modelling metapopulations with stochastic membrane systems. *Biosystems* **91**(3), 499–514 (2008)
10. D. Besozzi, P. Cazzaniga, D. Pescini, G. Mauri, S. Colombo, E. Martegani, The role of feedback control mechanisms on the establishment of oscillatory regimes in the Ras/cAMP/PKA pathway in *S. cerevisiae*. *EURASIP J. Bioinf. Syst. Biol.* **2012**, 10 (2012)
11. L. Bianco, A. Castellini, Psim: a computational platform for metabolic P systems, in *Workshop on Membrane Computing* (2007), pp. 1–20
12. BioSimWare. <http://biosimware.disco.unimib.it/>



13. J. Blakes, Infobiotics: computer-aided synthetic systems biology. Ph.D. thesis, School of Computer Science, University of Nottingham, UK, 2012
14. J. Blakes, J. Twycross, F.J. Romero-Campero, N. Krasnogor, The infobiotics workbench: an integrated in silico modelling platform for systems and synthetic biology. *Bioinformatics* **27**(23), 3323–3324 (2011)
15. M. Calder, A. Duguid, S. Gilmore, J. Hillston, Stronger computational modelling of signalling pathways using both continuous and discrete-state methods, in *Proceedings of CMSB 2006*, vol. 4210 (LNCS, 2006), pp. 63–77
16. M. Calder, S. Gilmore, J. Hillston, Automatically deriving ODEs from process algebra models of signalling pathways, in *Proceedings of CMSB 2005*, Edinburgh, Scotland (2005), pp. 204–215
17. M. Calder, S. Gilmore, J. Hillston, Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA, in *Transactions on Computational Systems Biology VII*, vol. 4230 of *Lecture Notes in Computer Science*, ed. by C. Priami, A. Inglsdttir, B. Mishra, H. Riis Nielson (Springer, Berlin Heidelberg, 2006), pp. 1–23
18. M. Calder, S. Gilmore, J. Hillston, V. Vyshemirsky, Formal methods for biochemical signalling pathways, in *Formal Methods: State of the Art and New Directions* (Springer, Berlin, 2006), pp. 185–215
19. B. Canton, A. Labno, D. Endy, Refine and standardization of synthetic biological parts and devices. *Nat. Biotechnol.* **26**(7), 787–793 (2008)
20. H. Cao, F.J. Romero-Campero, S. Heeb, M. Cámara, N. Krasnogor, Evolving cell models for systems and synthetic biology. *Syst. Synth. Biol.* **4**(1), 55–84 (2010)
21. Y. Cao, D.T. Gillespie, L.R. Petzold, Adaptive explicit–implicit tau-leaping method with automatic tau selection. *J. Chem. Phys.* **126**(22), 224101 (2007)
22. Y. Cao, L. Petzold, Slow scale tau-leaping method. *Comput. Methods Appl. Mech. Eng.* **197**, 43–44 (2008)
23. M. Cavaliere, T. Mazza, S. Sedwards, Statistical model checking of membrane systems with peripheral proteins: quantifying the role of estrogen in cellular mitosis and DNA damage, in *Applications of Membrane Systems to Biology, Emergence, Complexity and Computation*, ed. by P. Frisco, M. Gheorghe, M. Pérez-Jiménez (Springer, 2013)
24. P. Cazzaniga, D. Pescini, D. Besozzi, G. Mauri, Tau leaping stochastic simulation method in P systems, in *Workshop on Membrane Computing*, vol. 4361 (LNCS, 2006), pp. 298–313
25. F. Ciocchetta, J. Hillston, Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.* **410**(33–34), 3065–3084 (2009)
26. D. Corne, P. Frisco, Dynamics of HIV infection studied with cellular automata and conformon-P systems. *Biosystems* **3**(91), 531–544 (2008)
27. V. Danos, J. Feret, W. Fontana, J. Krivine, Scalable modelling of biological pathways, in *Asian Symposium on Programming Systems*, vol. 4807 (LNCS, 2007), pp. 139–157
28. M.A.M. del Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A.R.-N. Nez, F. Sancho-Caparrini, A formal verification algorithm for multienvironment probabilistic P systems. *Int. J. Found. Comput. Sci.* **22**(1), 107–118 (2011)
29. L. Dematté, C. Priami, A. Romanel, The beta workbench: a computational tool to study the dynamics of biological systems. *Brief. Bioinform.* **9**(5), 437–449 (2008)
30. L. Dematté, C. Priami, A. Romanel, The BlenX language: a tutorial, in *Formal Methods for Computational Systems Biology, SFM 2008*, vol. 5054 (LNCS, 2008), pp. 123–138
31. A. Deutsch, S. Dromann, *Cellular Automata Modeling of Biological Pattern Formation* (Springer, Heidelberg, 2009)
32. M. J. Dinneen, Y.-B. Kim, R. Nicolescu, Edge- and node-disjoint paths in P systems, in *Proceedings 4th Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2010)*, ed. by M.K.G. Ciobanu, vol. 40 (EPTCS, 2010), pp. 121–141
33. R. Donaldson, MC2(PLTLc) Monte Carlo Model Checker for PLTLc properties (2008)
34. R. Donaldson, D. Gilbert, A. Model checking approach, to the parameter estimation of biochemical pathways, in *CMSB 2008*, vol. 5307 (LNBI, Springer-Verlag, 2008), pp. 269–287

35. R. Donaldson, D. Gilbert, A Monte Carlo Model Checker for probabilistic LTL with numerical constraints. Technical report, Bioinformatics Research Centre, University of Glasgow, Glasgow, 2008
36. C. Dragomir, From P systems specification to prism. Master's thesis, Department of Computer Science, University of Sheffield, Sheffield, UK, 2009
37. M. B. Dwyer, G. S. Avrunin, J. C. Corbett, Patterns in property specifications for finite-state verification, in *Proceedings of the 21st international conference on Software engineering, ICSE '99* (ACM, 1999), pp. 411–420
38. J. Elf, M. Ehrenberg, Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Syst. Biol.* **1**(2), 230–236 (2004)
39. G.B. Ermentrout, L. Edelstein-Keshet, Cellular automata approaches to biology. *J. Theor. Biol.* **160**, 97–133 (1993)
40. J.R. Faeder, M.L. Blinov, W.S. Hlavacek, Rule-based modeling of biochemical systems with BioNetGen, in *Methods in Molecular Biology, Systems Biology*, vol. 500 of *Methods in Molecular Biology*, ed. by I.V. Maly (Humana Press, Totowa, 2009), pp. 113–167
41. A. Feiglin, A. Hacohen, A. Sarusi, J. Fisher, R. Unger, Y. Ofra, Static network structure can be used to model the phenotypic effects of perturbations in regulatory networks. *Bioinformatics* **28**(21), 2811–2818 (2012)
42. J. Fisher, T. Henzinger, Executable biology, in *Proceedings of the 2006 Winter Simulation Conference* (2006), pp. 1675–1682
43. J. Fisher, T.A. Henzinger, Executable cell biology. *Nat. Biotechnol.* **25**(11), 1239–1249 (2007)
44. J. Fisher, N. Piterman, E.J.A. Hubbard, M.J. Stern, D. Harel, Computational insights into *Caenorhabditis elegans* vulval development. *PNAS* **102**(6), 1951–1956 (2005)
45. D. Florine, J. Santiago, K. Betz, J. Twycross, S.-Y. Park, L. Rodriguez, M. Gonzalez-Guzman, M. Jensen, N. Krasnogor, M. Holdsworth, M. Blackledge, S. Cutler, P. Rodriguez, J. Marquez, A thermodynamic switch modulates abscisic acid receptor sensitivity. *EMBO J.* **30**, 4171–4184 (2011)
46. P. Frisco, *Computing with Cells: Advances in Membrane Computing* (Oxford University Press, Oxford, 2009)
47. A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, H. Kitano, Cell Designer 3.5: a versatile modeling tool for biochemical networks. *Proc. IEEE* **96**(8), 1254–1265 (2008)
48. M. Galassi, *GNU Scientific Library Reference Manual*, 3rd edn. (GNU, 2009)
49. Q. Gao, F. Liu, D. Tree, D. Gilbert, Multi-cell modelling using coloured Petri nets applied to planar cell polarity, in *Proceedings of the 2nd International Workshop on Biological Processes and Petri Nets (BioPPN2011)* (2011), pp. 135–150
50. L. Gerosa, Stochastic process algebras as design and analysis framework for synthetic biology modelling. Master's thesis, University of Trento, 2007
51. M. Gibson, J. Bruck, Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A* **104**(9), 1876–1889 (2000)
52. D. Gilbert, H. Fuss, X. Gu, R. Orton, S. Robinson, V. Vyshemirsky, M.J. Kurth, C.S. Downes, W. Dubitzky, Computational methodologies for modelling, analysis and simulation of signalling networks. *Brief. Bioinform.* **7**, 339–353 (2006)
53. D. Gilbert, M. Heiner, S. Lehrack, A unifying framework for modelling and analysing biochemical pathways using Petri nets, in *Proceedings of the, international conference on Computational methods in systems biology, CMSB'07* (Springer-Verlag, Berlin, 2007)
54. D. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.* **22**(4), 403–434 (1976)
55. D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977)
56. D.T. Gillespie, A rigorous derivation of the chemical master equation. *Physica A* **188**, 404–425 (1992)
57. D.T. Gillespie, Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.* **115**(4), 1716 (2001)

58. D.T. Gillespie, Stochastic simulation of chemical kinetics. *Ann. Rev. Phys. Chem.* **58**, 35–55 (2007)
59. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison Wesley, Boston, 1989)
60. L. Grunske, Specification patterns for probabilistic quality properties, in *Proceedings of the 30th international conference on Software engineering, ICSE '08* (ACM, 2008), pp. 31–40
61. M.L. Guerriero, D. Prandi, C. Priami, P. Quaglia, Process calculi abstractions for biology. Technical report, CoSBI (Center for Computational and Systems Biology), Trento, Italy, 2006
62. N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* **9**, 159–195 (2001)
63. H. Hansson, B. Jonsson, A logic for reasoning about time and reliability. *Formal Aspects Comput.* **6**, 102–111 (1994)
64. L.H. Hartwell, J.J. Hopfield, S. Leibler, A.W. Murray, From molecular to modular cell biology. *Nature* **402**, C47–C52 (1999)
65. M. Heiner, D. Gilbert, R. Donaldson, Petri nets for systems and synthetic biology. *Formal Methods Comput. Syst. Biol.* **5016**, 215–264 (2008)
66. T. Hinze, C. Bodenstern, B. Schau, I. Heiland, S. Schuster, Chemical analog computers for clock frequency control based on P modules, in *Proceedings of the 12th International Conference on Membrane Computing, CMC'11* (Springer-Verlag, 2012), pp. 182–202
67. T. Hinze, T. Lenser, G. Escuela, I. Heiland, S. Schuster, Modelling signalling networks with incomplete information about protein activation states: a P system framework for KaiABC oscillator, in *Workshop on Membrane Computing*, vol. 5957 (LNCS, 2010), pp. 316–334
68. S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, U. Kummer, COPASI—a Complex PATHway Simulator. *Bioinformatics* **22**(24), 3067–3074 (2006)
69. M. Hucka, A. Finney, J. Bornstein, M. Keating, E. Shapiro, J. Matthews, L. Kovitz, J. Schilstra, A. Funahashi, C. Doyle, H. Kitano, Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. *Syst. Biol.* **1**, 41–53 (2004)
70. J. Jack, A. Păun, Discrete modeling of biochemical signaling with memory enhancement. *Trans. Comput. Syst. Biol.* **11**, 200–215 (2009)
71. K. Jensen, L.M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems* (Springer, Berlin, 2009)
72. S.A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* **22**(3), 437–467 (1969)
73. S. Konrad, B. Cheng, Real-time specification patterns, in *Proceedings of 27th International Conference on Software Engineering* (2005), pp. 372–381
74. M. Kwiatkowska, J. Heath, E. Gaffney, Simulation and verification for computational modelling of signalling pathways, in *Proceedings of the 2006 Winter Simulation Conference* (2006), pp. 1666–1674
75. M. Kwiatkowska, G. Norman, D. Parker, Using probabilistic model checking in systems biology. *ACM Sigmetrics Perform. Eval. Rev.* **35**(4), 14–21 (2008)
76. M.Z. Kwiatkowska, G. Norman, D. Parker, Probabilistic model checking in practice: case studies with PRISM. *Sigmetrics Perform. Eval. Rev.* **32**(4), 16–21 (2005)
77. J. C. Lagarias, Point lattices, in *Handbook of Combinatorics*, vol. 1 (Elsevier, Amsterdam, 1995)
78. A. Mallavarapu, M. Thomson, B. Ullian, J. Gunawardena, Programming with models: modularity and abstraction provide powerful capabilities for systems biology. *J. R. Soc. Interface* **6**, 257–270 (2009)
79. V. Manca, The metabolic algorithm for P systems: principles and applications. *Theor. Comput. Sci.* **404**(1–2), 142–155 (2008)
80. V. Manca, Metabolic P systems. *Scholarpedia* **5**(3), 9273 (2010)
81. V. Manca, *Infobiotics: Information in Biotic Systems* (Springer, Berlin, 2013)

82. V. Manca, L. Marchetti, Log-gain stoichiometric stepwise regression for MP systems. *Int. J. Found. Comput. Sci.* **22**(01), 97–106 (2011)
83. M. Martínez-del Amor, I. Pérez-Hurtado, A. Gastalver-Rubio, A. Elster, M. Pérez-Jiménez, Population dynamic P systems on CUDA, in *Workshop on Membrane Computing*, vol. 7605 (LNCS, 2012), pp. 247–266–313
84. Matplotlib. <http://matplotlib.org>
85. MetaPlab. <http://mplab.sci.univr.it/>
86. R. Milner, *Communicating and Mobile Systems:  $\pi$ -Calculus* (Cambridge University Press, Cambridge, 1999)
87. I.I. Moraru, J.C. Schaff, B.M. Slepchenko, L.L.M. The, virtual cell: an integrated modeling environment for experimental and computational cell biology. *Ann. N. Y. Acad. Sci.* **971**, 595–596 (2002)
88. S. Natkin, Les Réseaux de Petri Stochastiques et leur Application a l'Evaluation des Systemes Informatiques. Ph.D. thesis, CNAM, Paris, France, 1980
89. A. Obdulowicz, Generalized Gandy–Păun–Rozenberg machines for tile systems and cellular automata, in *Membrane Computing*, vol. 7184 of *Lecture Notes in Computer Science*, ed. by M. Gheorghe, G. Păun, G. Rozenberg, A. Salomaa, S. Verlan (Springer, Berlin Heidelberg, 2012), pp. 314–332
90. Z. Ognjanović, Discrete linear-time probabilistic logics: completeness, decidability and complexity. *J. Logic Comput.* **16**(2), 257–285 (2006)
91. G. Păun, Computing with membranes. *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000)
92. G. Păun, F.J. Romero-Campero, Membrane computing as a modeling framework. Cellular systems case studies, in *Formal Methods for Computational Systems Biology*, vol. 5016 (LNCS, 2008), pp. 168–214
93. M. Pedersen, A. Phillips, Towards programming languages for genetic engineering of living cells. *J. R. Soc. Interface R. Soc.* **6**(Suppl 4), S437–S450 (2009)
94. M. Pedersen, G. D. Plotkin, A language for biochemical systems : design and formal specification, in *Transactions on Computational Systems Biology XIIC*, ed. by C. Priami, vol. 5945 (LNBI, 2010), pp. 77–145
95. I. Pérez-Hurtado, L. Valencia, M. Pérez-Jiménez, M. Colomer, A. Riscos-Núñez, General purpose software tool for simulating biological phenomena by means of P Systems, in *Proceedings of IEEE 5th International Conference BIC-TA 2010*, Changsha, China (2010), pp. 637–643
96. D. Pescini, D. Besozzi, G. Mauri, C. Zandron, Dynamic probabilistic P systems. *Int. J. Found. Comput. Sci.* **1**(17), 183–204 (2006)
97. Petri nets tool database. <http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html>
98. A. Phillips, L. Cardelli, A correct abstract machine for the stochastic Pi-calculus, in *Concurrent Models in Molecular Biology, BioConcur '04* (ENTCS, 2004)
99. A. Phillips, L. Cardelli, G. Castagna, A graphical representation for biological processes in the stochastic  $\pi$ -calculus. *Trans. Comput. Syst. Biol.* **4230**, 123–152 (2006)
100. A. Pnueli, The temporal logic of programs, in *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science* (IEEE Computer Society Press, 1977), pp. 46–57
101. C. Priami, Stochastic  $\pi$ -calculus. *Comput. J.* **38**(7), 578–589 (1995)
102. C. Priami, Algorithmic systems biology. *Commun. ACM* **52**, 80–88 (2009)
103. C. Priami, P. Quaglia, Modelling the dynamics of biosystems. *Brief. Bioinform.* **5**(3), 259–269 (2004)
104. C. Priami, A. Regev, E. Shapiro, W. Silverman, Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inform. Process. Lett.* **80**, 25–31 (2001)
105. Pulse Generator. <http://www.infobiotic.org/models/pulseGenerator/pulseGenerator.html>
106. R. Ramaswamy, N. González-Segredo, I.F. Sbalzarini, A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks. *J. Chem. Phys.* **130**(24), 244104 (2009)

107. V.N. Reddy, M.N. Liebman, M.L. Mavrovouniotis, Qualitative analysis of biochemical reaction systems. *Comput. Biol. Med.* **26**, 9–24 (1996)
108. V.N. Reddy, M.L. Mavrovouniotis, M.N. Liebman, Petri net representations in metabolic pathways. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* **1**, 328–336 (1993)
109. A. Regev, W. Silverman, E. Shapiro, Representation and simulation of biochemical processes using the  $\pi$ -calculus process algebra, in *Pacific Symposium on Biocomputation*, vol. 26 (2001), pp. 459–470
110. F. Romero-Campero, M. Gheorghe, L. Bianco, D. Pescini, M. Pérez-Jiménez, R. Ceterchi, Towards probabilistic model checking on P systems using PRISM, in *Membrane Computing*, volume 4361 (LNCS/Springer, Berlin, 2006), pp. 477–495
111. F. J. Romero-Campero, H. Cao, M. Camara, N. Krasnogor, Structure and parameter estimation for cell systems biology models, in *Proceedings of the 10th annual conference on Genetic and Evolutionary Computation (GECCO '08)* (2008), pp. 331–339
112. F.J. Romero-Campero, M. Gheorghe, G. Ciobanu, J.M. Auld, M.J. Pérez-Jiménez, Cellular modelling using P systems and process algebra. *Prog. Nat. Sci.* **17**, 375–383 (2007)
113. F.J. Romero-Campero, J. Twycross, M. Cámara, M. Bennett, M. Gheorghe, N. Krasnogor, Modular assembly of cell systems biology models using P systems. *Int. J. Found. Comput. Sci.* **20**(03), 427–442 (2009)
114. F. J. Romero-Campero, J. Twycross, H. Cao, J. Blakes, N. Krasnogor, A multiscale modeling framework based on P systems, in *WMC9 2008* (Springer-Verlag, Berlin, 2009), pp. 63–77
115. The p-lingua web page. <http://www.p-lingua.org>
116. R.P. Shetty, D. Endy, T.F. Knight, Engineering BioBrick vectors from BioBrick parts. *J. Biol. Eng.* **2**, 5 (2008)
117. A. Slepoy, A.P. Thompson, S.J. Plimpton, A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks. *J. Chem. Phys.* **128**(20), 205101 (2008)
118. M.W. Sneddon, J.R. Faeder, T. Emonet, Efficient modeling, simulation and coarse-grain of biological complexity with NFsim. *Nat. Methods* **8**, 177–183 (2011)
119. R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**, 341–359 (1997)
120. F. J. W. Symons, Introduction to numerical Petri nets, a general graphical model of concurrent processing systems. *Aust. Telecommun. Res.* **14**(1), 26 (1980)
121. J. Twycross, L.R. Band, M.J. Bennett, J.R. King, N. Krasnogor, Stochastic and deterministic multiscale models for systems biology: an auxin-transport case study. *BMC Syst. Biol.* **4**(1), 1–34 (2010)
122. S. Verlan, F. Bernardini, M. Gheorghe, M. Margenstern, Generalized communicating P systems. *Theor. Comput. Sci.* **404**(1–2), 170–184 (2008)
123. J. Will, M. Heiner, Petri nets in biology, chemistry, and medicine. Bibliography. Technical report, Brandenburg University of Technology at Cottbus, 2002
124. S. Wolfram, *A New Kind of Science* (Wolfram Media, Champaign, 2002)
125. A. Yachie-Kinoshita, T. Nishino, H. Shimo, M. Suematsu, M. Tomita, A metabolic model of human erythrocytes: practical application of the e-cell simulation environment. *Biomed. Biotechnol.* **2010**, 642420 (2010)

## Chapter 2

# Statistical Model Checking of Membrane Systems with Peripheral Proteins: Quantifying the Role of Estrogen in Cellular Mitosis and DNA Damage

Matteo Cavaliere, Tommaso Mazza and Sean Sedwards

**Abstract** Systems biology is a natural application of membrane systems, allowing the analysis of biological systems using the formal technique of model checking. To overcome the intractable model size of typical biological systems, *statistical* model checking may be used to efficiently estimate the probability of properties of interest with arbitrary levels of confidence. In this chapter we analyse a biological system linked to breast cancer, using statistical model checking (SMC) applied to membrane systems. To do this, we have constructed a computational platform that integrates an SMC library with a stochastic simulator of membrane systems with peripheral proteins. We present the methodology to investigate the role of estrogen in cellular mitosis and DNA damage and we use our statistical model checker to find the most appropriate time-dependent dosage of antagonist that should be used to minimize the uncontrolled replication of abnormal cells.

## 2.1 Membrane Systems with Peripheral Proteins

Membrane systems are models of computation inspired by the structure and function of biological cells. The model was introduced in 1998 by Gh. Păun and since then many classes of membrane systems have been introduced and studied, with mathematical, computer science and biological motivations. An introductory guide to the

---

M. Cavaliere (✉)

Spanish National Center for Biotechnology, Madrid, Spain  
e-mail: mcavaliere@cnb.csic.es

T. Mazza

IRCCS Casa Sollievo della Sofferenza—Mendel laboratory, Rome, Italy  
e-mail: t.mazza@css-mendel.it

S. Sedwards

INRIA Rennes—Bretagne Atlantique, Rennes, France  
e-mail: sean.sedwards@inria.fr



field can be found in the recent handbook [1], while a review of applications of membrane computing to biology can be found in [2].

According to the original definition [3], membrane systems comprise an hierarchical nesting of membranes that enclose regions (representing the *cellular structure*), in which free-floating objects (representing *molecules*) exist. Each region can have associated rules, called *evolution rules*, for evolving the free-floating objects and modelling the biochemical reactions present in cellular compartments. Rules also exist for moving objects in a synchronized manner across membranes, *symport* and *antiport* rules, modelling cellular transport and more general communication rules [1].

In brane calculi, presented in [4], several operations (*pino*, *exo*, *phago*, *mate*, *drip*, *bud*) explicitly involving membranes with embedded proteins are considered and formalised in the framework of process calculi. An important difference between brane calculi and membrane computing is that with brane calculi the evolution of the system takes place *on* the membranes and not inside the compartments delimited by them. In [5] the operations of brane calculi are represented in the membrane computing framework and then studied by using tools from formal language theory. In [6] some of the membrane operations (pinocytosis and dripping) are considered in combination with the presence of free-floating objects and objects attached to the membranes, while in [7] objects (peripheral proteins) are attached to either side of a membrane, explicitly considering the inner and outer membrane surfaces. The motivation for this last model is to represent the cellular processes that are controlled by the presence of specific proteins on the appropriate side of and integral to the membrane: there is a constant interaction between floating chemicals and embedded proteins and between peripheral and integral proteins [8]. Essential receptor-mediated processes, such as endocytosis and signalling, are crucial to cell function and by definition are critically dependent on the presence of peripheral and integral membrane proteins.

The key features of the model considered in [7] are that in each region of the system there are floating objects (the floating chemicals) and, in addition, objects can be associated to each side of a membrane or integral to the membrane (the peripheral and integral membrane proteins). Systems constructed using this model can perform the following operations: (i) the floating objects can be processed/changed inside the regions of the system (emulating biochemical reactions) and (ii) the floating and attached objects can be processed/changed when they interact (modelling the interactions of the floating molecules with membrane proteins). A possible use of the model to study biological processes is shown in [7, 9], while related models are discussed in [10] and [11], where also the computational aspects are presented.

The use of a formal computational model such as membrane systems can be helpful for two reasons: to facilitate the implementation of an executable specification and allows the use of automatic methods to analyse and to discover features concerning the dynamics of the complex cellular systems, providing in this way algorithms that can mimic biological phenomena, [12].

In this book chapter we approach this second possibility by presenting model checking, an algorithmic technique to formally verify the performance of a system with respect to a property. The system is represented in a language with formal

semantics (in our case, multiset rewriting) and the property is expressed in temporal logics (e.g., *linear temporal logic*, LTL, and *computation tree logic*, CTL). The output of standard model checking algorithms for these logics is Boolean: the model either satisfies the property or it does not. Such algorithms have polynomial time complexity with respect to the size of the model, but this is generally exponentially related to the description of the model (i.e., the number of interacting components). See [13] for a recent historical overview of standard model checking techniques and [14] for comprehensive coverage.

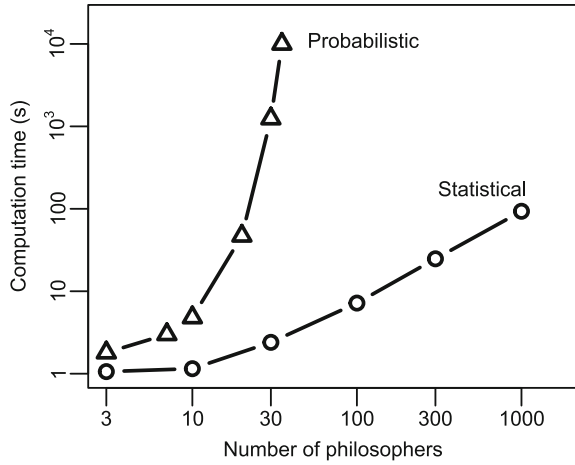
Many real systems are modelled using some form of non-determinism to account for unknown interactions and environments. In particular, chemical systems are often modelled with the implicit assumption that molecules move randomly, with probabilities of interaction proportional to the total number of molecules (so-called *mass action* [15]). When non-determinism is expressed with probabilities, it is possible to quantify the probability of a property using *probabilistic* model checking [16], which uses probabilistic or stochastic temporal logics (e.g. *probabilistic computation tree logic*, PCTL, and *continuous stochastic logic*, CSL) and numerical techniques to calculate the probability of a property. Probabilistic model-checking algorithms use the standard model checking algorithms to evaluate whether a formula is satisfied in a particular state, but incur additional computational cost (polynomial w.r.t. the model) to calculate the probability of being in the state. Although techniques and data structures exist to minimise the model [13, 14, 16], in the majority of real applications (especially biological applications) the model remains intractable. Notable successes of standard and probabilistic model checking applied to simplified biological systems include [12, 17–19]. In particular, in [17] the authors have shown the use of probabilistic model checking for the analysis of the cell cycle in eukaryotes, using a modelling language based on membrane systems and process algebra. Platforms based on membrane systems including model checking have been previously implemented, [20]. A review of the probabilistic models in membrane systems can be found in [1].

To overcome the state space explosion problem that afflicts most biological models, in this chapter we employ statistical model checking (SMC), which is an efficient, approximative, variety of probabilistic model checking. SMC has been applied to biology before (e.g., [21]), but here we present the first SMC investigation of a biological system using a membrane systems model that explicitly considers the role of membrane proteins. To achieve this, we present the first self-contained statistical model checker dedicated to membrane systems with peripheral and integral proteins.

There are important differences between probabilistic and statistical model checking. The characteristic feature of a statistical model checker is that it estimates the probability of a property by verifying the property against multiple independent executions (simulations) of the system. The confidence of the estimate can be guaranteed to arbitrary levels of confidence by standard statistical bounds (e.g. the Chernoff bound [22]) and in this way SMC trades certainty for tractability. In comparison to standard and probabilistic model checking, SMC does not require a finite state space, does not require decidable logics and is less strict about how the system is defined. Importantly, SMC is often significantly more efficient than probabilistic



**Fig. 2.1** Typical performance of probabilistic and statistical model checking applied to models of the probabilistic dining philosophers protocol



model checking for a given level of precision. Figure 2.1 compares the performance of probabilistic with statistical model checking applied to increasing size models of the probabilistic dining philosophers protocol, considering the property that if a philosopher is hungry, he will eventually be fed. The figure shows that the numerical model checker [23] scales exponentially with increasing numbers of philosophers (i.e., polynomially w.r.t. a model that increases exponentially), while the statistical model checker [24] scales linearly (proportional to the length of the property).

Further details of our SMC methodology are given in Sect. 2.2.1.

### 2.1.1 Formal Language Preliminaries

Membrane systems are based on *formal language theory* and *multiset rewriting* [25]. In this section we recall the theoretical notions and notations necessary in this chapter.

Given the set  $A$  we denote by  $|A|$  its cardinality and by  $\emptyset$  the empty set. We denote by  $\mathbb{N}$  and by  $\mathbb{R}$  the set of natural and real numbers, respectively.

As usual, an *alphabet*  $V$  is a finite set of symbols. By  $V^*$  we denote the set of all strings over  $V$ . By  $V^+$  we denote the set of all strings over  $V$  excluding the empty string. The empty string is denoted by  $\lambda$ . The *length* of a string  $v$  is denoted by  $|v|$ . The concatenation of two strings  $u, v \in V^*$  is written  $uv$ .

The number of occurrences of the symbol  $a$  in the string  $w$  is denoted by  $|w|_a$ .

A *multiset* is a set where each element may have a multiplicity. Formally, a multiset over a set  $V$  is a map  $M : V \rightarrow \mathbb{N}$ , where  $M(a)$  denotes the multiplicity of the symbol  $a \in V$  in the multiset  $M$ .

For multisets  $M$  and  $M'$  over  $V$ , we say that  $M$  is *included in*  $M'$  if  $M(a) \leq M'(a)$  for all  $a \in V$ . Every multiset includes the *empty multiset*, defined as  $M$  where  $M(a) = 0$  for all  $a \in V$ .

The *sum* of multisets  $M$  and  $M'$  over  $V$  is written as the multiset  $(M + M')$ , defined by  $(M + M')(a) = M(a) + M'(a)$  for all  $a \in V$ . The *difference* between  $M$  and  $M'$  is written as  $(M - M')$  and defined by  $(M - M')(a) = \max\{0, M(a) - M'(a)\}$  for all  $a \in V$ . We also say that  $(M + M')$  is obtained by *adding*  $M$  to  $M'$  (or vice versa) while  $(M - M')$  is obtained by *removing*  $M'$  from  $M$ . For example, given the multisets  $M = \{a, b, b, b\}$  and  $M' = \{b, b\}$ , we can say that  $M'$  is included in  $M$ , that  $(M + M') = \{a, b, b, b, b, b\}$  and that  $(M - M') = \{a, b\}$ .

If the set  $V$  is finite, e.g.  $V = \{a_1, \dots, a_n\}$ , then the multiset  $M$  can be explicitly described as  $\{(a_1, M(a_1)), (a_2, M(a_2)), \dots, (a_n, M(a_n))\}$ . The *support* of a multiset  $M$  is defined as the set  $\text{supp}(M) = \{a \in V \mid M(a) > 0\}$ . A multiset is empty (hence finite) when its support is empty (also finite).

A compact string notation can be used for finite multisets: if  $M = \{(a_1, M(a_1)), (a_2, M(a_2)), \dots, (a_n, M(a_n))\}$  is a multiset of finite support, then the string  $w = a_1^{M(a_1)} a_2^{M(a_2)} \dots a_n^{M(a_n)}$  (and all its permutations) precisely identify the symbols in  $M$  and their multiplicities. Hence, given a string  $w \in V^*$ , we can say that it identifies a finite multiset over  $V$ , written as  $M(w)$ , where  $M(w) = \{a \in V \mid (a, |w|_a)\}$ . For instance, the string  $bab$  represents the multiset  $M(w) = \{(a, 1), (b, 2)\}$ , that is the multiset  $\{a, b, b\}$ . The empty multiset is represented by the empty string  $\lambda$ .

### 2.1.2 Membrane Systems with Peripheral and Integral Proteins

Formal language theory can be used to provide a mathematical abstraction for the bidirectional interactions of floating molecules with cell membranes: biochemical rules and interactions between peripheral proteins and membranes can be formalised in terms of multiset rewriting rules.

In this section we introduce the main notions for membrane systems with peripheral and integral proteins.

As it is usual in the membrane systems field, a membrane is represented by a pair of square brackets,  $[ ]$ . A *membrane structure* is an *hierarchical nesting* of membranes enclosed by a main membrane called the *root membrane*. A label is associated to each membrane and it is written as a superscript of the membrane, e.g.  $[ ]^1$ . If a membrane has the label  $i$  we call it membrane  $i$ . Each membrane is identified by a unique label in a unique manner (there are no membranes with the same label).

A membrane structure is essentially that of a tree data structure, where the nodes are the membranes and the arcs represent the containment relation. Being a tree, a membrane structure can be represented by a string of matching square brackets, e.g.,  $[ [ [ ]^2 ]^1 [ ]^3 ]^0$ .

To each membrane there are associated three multisets,  $u$ ,  $v$  and  $x$  over  $V$ , denoted by  $[ ]_{u|v|x}$ , where  $V$  denotes a finite alphabet of *objects* (the symbol  $|$  is not part of the alphabet  $V$ ).

Following the terminology used in [9] we say that the membrane is *marked* by  $u$ ,  $v$  and  $x$ ;  $x$  is called the *external marking*,  $u$  the *internal marking* and  $v$  the *integral marking* of the membrane. In general, we refer to them as *markings* of the membrane.

The internal, external and integral markings of a membrane model the proteins attached to the internal surface, to the external surface and integral to the membrane, respectively.

In a membrane structure, the region between membrane  $i$  and any enclosed membranes is called region  $i$ . To each region is associated a multiset of objects  $w$  called the *free objects* of the region. The free objects are written between the brackets enclosing the regions, e.g.,  $[aa [bb]^1]^0$ . The free objects of a membrane model the floating chemicals within the regions of a cell.

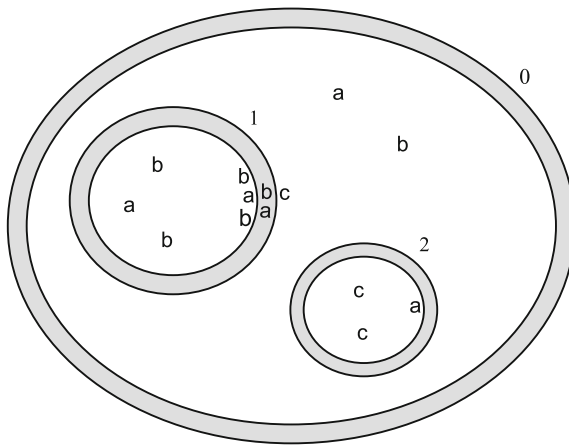
We denote by  $int(i)$ ,  $ext(i)$  and  $itgl(i)$  the internal, external and integral markings of membrane  $i$ , respectively. By  $free(i)$  we denote the free objects of region  $i$ . For any membrane  $i$ , distinct from a root membrane, we denote by  $out(i)$  the label of the membrane enclosing membrane  $i$ . The finite set of all possible labels is denoted by  $Lab$ .

The string

$$[ab [cc]_{a|}^2 [abb]_{bba|ab|c}^1]^0$$

represents, for instance, a membrane structure, where to each membrane are associated markings and to each region are associated free objects. Membrane 1 is internally marked by  $bba$  (i.e.,  $int(1) = bba$ ), has integral marking  $ab$  (i.e.,  $itgl(1) = ab$ ) and is externally marked by  $c$  (i.e.,  $ext(1) = c$ ). To region 1 are associated the free objects  $abb$  (i.e.,  $free(1) = abb$ ). To region 0 are associated the free objects  $ab$ . Finally,  $out(1) = out(2) = 0$ . Membrane 0 is the root membrane. The string can also be depicted diagrammatically, as in Fig. 2.2.

As in [9] we consider the rules  $attach_{in}$ ,  $attach_{out}$ ,  $de-attach_{in}$  and  $de-attach_{out}$ , defined in the following manner:



**Fig. 2.2** Graphical representation of  $[ab [cc]_{a|}^2 [abb]_{bba|ab|c}^1]^0$

$$\begin{aligned}
attach_{in} &: [\alpha]_{u|v}^i \rightarrow [\ ]_{u'|v'}^i, \quad \alpha \in V^+, u, v, u', v' \in V^*, i \in Lab \\
attach_{out} &: [\ ]_{v|x}^i \alpha \rightarrow [\ ]_{v'|x'}^i, \quad \alpha \in V^+, v, x, v', x' \in V^*, i \in Lab \\
de-attach_{in} &: [\ ]_{u|v}^i \rightarrow [\alpha]_{u'|v'}^i, \quad \alpha, u', v', u, v \in V^*, |uv| > 0, i \in Lab \\
de-attach_{out} &: [\ ]_{v|x}^i \rightarrow [\ ]_{v'|x'}^i \alpha, \quad \alpha, v', x', v, x \in V^*, |vx| > 0, i \in Lab
\end{aligned}$$

Using the notion of multiset presented earlier, we describe the formal semantics of the rules.

The  $attach_{in}$  rule is *applicable to membrane  $i$*  if  $free(i)$  includes  $\alpha$ ,  $int(i)$  includes  $u$  and  $itgl(i)$  includes  $v$ . When the rule is *applied to membrane  $i$* ,  $\alpha$  is removed from  $free(i)$ ,  $u$  is removed from  $int(i)$ ,  $v$  is removed from  $itgl(i)$ ,  $u'$  is added to  $int(i)$  and  $v'$  is added to  $itgl(i)$ . The objects not involved in the application of the rule are left unchanged in their original positions.

The  $attach_{out}$  rule is applicable to membrane  $i$  if  $free(out(i))$  includes  $\alpha$ ,  $itgl(i)$  includes  $v$ ,  $ext(i)$  includes  $x$ . When the rule is applied to membrane  $i$ ,  $\alpha$  is removed from  $free(out(i))$ ,  $v$  is removed from  $itgl(i)$ ,  $x$  is removed from  $ext(i)$ ,  $v'$  is added to  $itgl(i)$  and  $x'$  is added to  $ext(i)$ . The objects not involved in the application of the rule are left unchanged in their original positions.

The  $de-attach_{in}$  rule is applicable to membrane  $i$  if  $int(i)$  includes  $u$  and  $itgl(i)$  includes  $v$ . When the rule is applied to membrane  $i$ ,  $u$  is removed from  $int(i)$ ,  $v$  is removed from  $itgl(i)$ ,  $u'$  is added to  $int(i)$ ,  $v'$  is added to  $itgl(i)$  and  $\alpha$  is added to  $free(i)$ . The objects not involved in the application of the rule are left unchanged in their original positions.

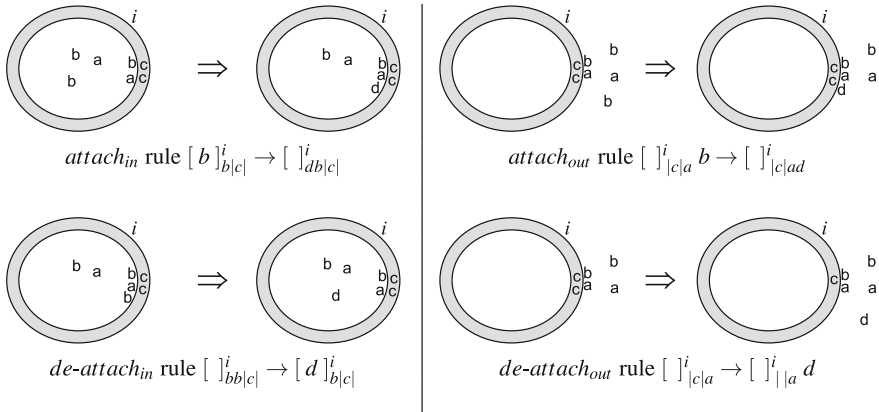
The  $de-attach_{out}$  rule is applicable to membrane  $i$  if  $itgl(i)$  includes  $v$  and  $ext(i)$  includes  $x$ . When the rule is applied to membrane  $i$ ,  $v$  is removed from  $itgl(i)$ ,  $x$  is removed from  $ext(i)$ ,  $v'$  is added to  $itgl(i)$ ,  $x'$  is added to  $ext(i)$  and  $\alpha$  is added to  $free(out(i))$ . The objects not involved in the application of the rule are left unchanged in their original positions.

Instances of  $attach_{in}$ ,  $attach_{out}$ ,  $de-attach_{in}$  and  $de-attach_{out}$  rules are depicted in Fig. 2.3.

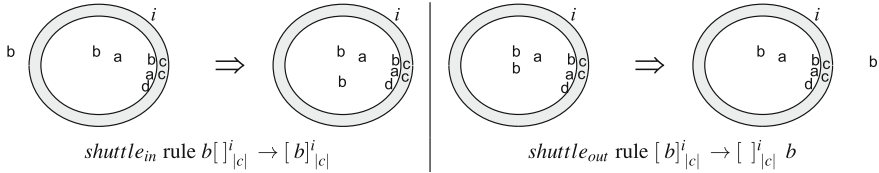
Extending the model in [9], we also consider rules that model the shuttling and translocation of proteins across membranes, as those considered in [10, 11]. In this case, floating objects can cross membranes depending on the proteins present on the membrane (proteins may also change during the translocation) Fig. 2.4.

$$\begin{aligned}
shuttle_{out} &: [\alpha]_{u|v}^i \rightarrow [\ ]_{u'|v'}^i \alpha, \quad \alpha \in V^+, u, v, u', v' \in V^*, i \in Lab \\
shuttle_{in} &: [\ ]_{v|x}^i \alpha \rightarrow [\alpha]_{v'|x'}^i, \quad \alpha \in V^+, v, x, v', x' \in V^*, i \in Lab
\end{aligned}$$

The operation of translocation and shuttling can be envisaged as an instantaneous combination of attach and de-attach rules described above. The  $shuttle_{in}$  rule is applicable to membrane  $i$  if  $itgl(i)$  includes  $v$ ,  $ext(i)$  includes  $x$  and  $free(out(i))$  includes  $\alpha$ . When the rule is applied to membrane  $i$ ,  $v$  is removed from  $itgl(i)$ ,  $x$  is



**Fig. 2.3** Examples of  $\text{attach}_{in}$ ,  $\text{attach}_{out}$ ,  $\text{de-attach}_{in}$  and  $\text{de-attach}_{out}$  rules, showing how free and attached objects may be rewritten. For example, in the  $\text{attach}_{in}$  rule one of the two free instances of  $b$  is rewritten to  $d$  and added to the membrane's internal marking



**Fig. 2.4** Examples of  $\text{shuttle}_{in}$  and  $\text{shuttle}_{out}$  rules, showing how free objects can cross membranes with the help of peripheral and integral proteins

removed from  $\text{ext}(i)$ ,  $v'$  is added to  $\text{itgl}(i)$ ,  $x'$  is added to  $\text{ext}(i)$  and  $\alpha$  is added to  $\text{free}(i)$ . The objects not involved in the application of the rule are left unchanged in their original positions.

The  $\text{shuttle}_{out}$  rule is applicable to membrane  $i$  if  $\text{itgl}(i)$  includes  $v$ ,  $\text{int}(i)$  includes  $u$  and  $\text{free}(i)$  includes  $\alpha$ . When the rule is applied to membrane  $i$ ,  $v$  is removed from  $\text{itgl}(i)$ ,  $u$  is removed from  $\text{int}(i)$ ,  $v'$  is added to  $\text{itgl}(i)$ ,  $u'$  is added to  $\text{int}(i)$  and  $\alpha$  is added to  $\text{free}(out(i))$ . The objects not involved in the application of the rule are left unchanged in their original positions.

We denote by  $\mathcal{R}_{V,Lab}^{att}$  the set of all possible  $\text{attach}$ ,  $\text{de-attach}$  and  $\text{shuttle}$  rules over the alphabet  $V$  and set of labels  $Lab$ .

As in [9], we also consider *evolution rules* that replace the free objects contained in a region conditional on the markings of the enclosing membrane. These rules represent the biochemical reactions that take place within the cytoplasm of a cell. An evolution rule has the following syntax:

$$\text{evol} : [ \alpha \rightarrow \beta ]_{u|v}^i$$

where  $u, v, \beta \in V^*$ ,  $\alpha \in V^+$ , and  $i \in Lab$ .

The semantics of the rule is as follows. The rule is applicable to region  $i$  if  $free(i)$  includes  $\alpha$ ,  $int(i)$  includes  $u$  and  $itgl(i)$  includes  $v$ . When the rule is applied to region  $i$ ,  $\alpha$  is removed from  $free(i)$  and  $\beta$  is added to  $free(i)$ . The membrane markings and the objects not involved in the application of the rule are left unchanged in their original positions.

We denote by  $\mathcal{R}_{V,Lab}^{ev}$  the set of all evolution rules over the alphabet  $V$  and set of labels  $Lab$ . An instance of an evolution rule is represented in Fig. 2.5.

In general, when a rule has label  $i$  we say that a rule is associated to *membrane*  $i$  (in the case of *attach* and *de-attach* rules) or is associated to *region*  $i$  (in the case of *evol* rules). For instance, in Fig. 2.3 the *attach<sub>in</sub>* is associated to membrane  $i$ .

The objects of  $\alpha$ ,  $u$  and  $v$  for *attach<sub>in</sub>/evol* rules, of  $\alpha$ ,  $v$  and  $x$  for *attach<sub>out</sub>* rules, of  $u$  and  $v$  for *de-attach<sub>in</sub>* rules and of  $v$  and  $x$  for *de-attach<sub>out</sub>* rules are the *reactants* of the corresponding rules. E.g., in the *attach* rule  $[b]_{a|c} \rightarrow [d]_{c|}$ , the reactants are  $a$ ,  $b$  and  $c$ .

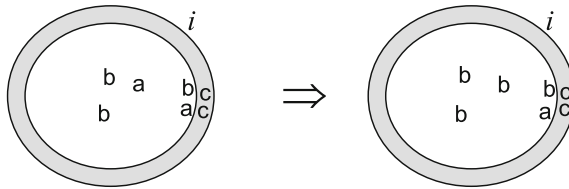
A membrane system with peripheral and integral proteins is a mathematical model that considers membranes to which can be associated peripheral proteins, integral proteins, free objects and using the operations described above. The rules presented here can be implemented in the computational tool using the appropriate syntax [26], and present redundancies whose purpose is to allow flexibility during the modelling processes. Several specific restricted variants of the proposed rules have been investigated and a review can be found in [10].

Here, following [9], we consider the stochastic extension of the model.

**Definition 1** A *stochastic membrane system with peripheral and integral proteins* and  $n$  membranes is a construct

$$\Pi = (V_\Pi, \mu_\Pi, (u_0, v_0, x_0)_\Pi, \dots, (u_{n-1}, v_{n-1}, x_{n-1})_\Pi, w_{0,\Pi}, \dots, w_{n-1,\Pi}, R_\Pi, t_{in,\Pi}, t_{fin,\Pi}, rate_\Pi)$$

- $V_\Pi$  is a non-empty alphabet of objects.
- $\mu_\Pi$  is a membrane structure with  $n \geq 1$  membranes injectively labelled by labels in  $Lab_\Pi = \{0, 1, \dots, n-1\}$ , where 0 is the label of the root membrane.
- $(u_0, v_0, x_0)_\Pi = (\lambda, \lambda, \lambda), (u_1, v_1, x_1)_\Pi, \dots, (u_{n-1}, v_{n-1}, x_{n-1})_\Pi \in V^* \times V^* \times V^*$  are called *initial markings of the membranes*.



**Fig. 2.5** *evol* rule  $[a \rightarrow b]_{b|c}^i$ . Free objects can be rewritten inside the region and the rewriting can depend on the integral and internal markings of the enclosing membrane

- $w_{0,\Pi}, w_{1,\Pi}, \dots, w_{n-1,\Pi} \in V^*$  are called *initial free objects of the regions*.
- $R_\Pi \subseteq \mathcal{R}_{V, Lab_\Pi - \{0\}}^{att} \cup \mathcal{R}_{V, Lab_\Pi}^{ev}$  is a finite set of evolution rules, attach/de-attach and shuttle rules.
- $t_{in,\Pi}, t_{fin,\Pi} \in \mathbb{R}$  are called the *initial time* and the *final time*, respectively.
- $rate_\Pi : R_\Pi \mapsto \mathbb{R}$  is the *rate mapping*. It associates to each rule a *reaction rate*.

An *instantaneous description*  $I$  of  $\Pi$  consists of the membrane structure  $\mu_\Pi$  with markings associated to the membranes and free objects associated to the regions. We denote by  $\mathbb{I}(\Pi)$  the set of *all instantaneous descriptions* of  $\Pi$ . We say *membrane (region)  $i$  of  $I$*  to denote the membrane (region, respectively)  $i$  present in  $I$ .

Let  $I$  be an arbitrary instantaneous description from  $\mathbb{I}(\Pi)$  and  $r$  an arbitrary rule from  $R_\Pi$ . Suppose that  $r$  is associated to membrane  $i \in Lab_\Pi$  if  $r \in \mathcal{R}_{V, Lab_\Pi - \{0\}}^{att}$  (or to region  $i \in Lab_\Pi$  if  $r \in \mathcal{R}_{V, Lab_\Pi}^{ev}$ ).

If  $r$  is applicable to membrane  $i$  (or to region  $i$ , accordingly) of  $I$ , then we say that  $r$  is *applicable to  $I$* . We denote by  $r(I) \in \mathbb{I}(\Pi)$  the instantaneous description of  $\Pi$  obtained when the rule  $r$  is applied to membrane  $i$  (or to region  $i$ , accordingly) of  $I$  (in short, we say  $r$  is *applied to  $I$* ).

The *initial instantaneous description* of  $\Pi$ ,  $I_{in,\Pi} \in \mathbb{I}(\Pi)$ , consists of the membrane structure  $\mu_\Pi$  with membrane  $i$  marked by  $(u_i, v_i, x_i)_\Pi$  for all  $i \in Lab_\Pi - \{0\}$  and free objects  $w_{i,\Pi}$  associated to region  $i$  for all  $i \in Lab_\Pi$ .

A *configuration* of  $\Pi$  is a pair  $(I, t)$  where  $I \in \mathbb{I}(\Pi)$  and  $t \in \mathbb{R}$ ;  $t$  is called the *time* of the configuration. We denote by  $\mathcal{C}(\Pi)$  the set of all configurations of  $\Pi$ . The *initial configuration* of  $\Pi$  is  $C_{in,\Pi} = (I_{in,\Pi}, t_{in,\Pi})$ .

Suppose that  $R_\Pi = \{rule^1, rule^2, \dots, rule^m\}$  and let  $S$  be an arbitrary sequence of configurations  $\langle C_0, C_1, \dots, C_j, C_{j+1}, \dots, C_h \rangle$ , where  $C_j = (I_j, t_j) \in \mathcal{C}(\Pi)$  for  $0 \leq j \leq h$ . Let  $a_j = \sum_{i=1}^m p_j^i$ ,  $0 \leq j \leq h$ , where  $p_j^i$  is the product of  $rate(rule^i)$

and the *mass action* combinatorial factor [27] for  $rule^i$  and  $I_j$ .

The sequence  $S$  is an *evolution* of  $\Pi$  if

- for  $j = 0$ ,  $C_j = C_{in,\Pi}$
- for  $0 \leq j \leq h - 1$ ,  $a_j > 0$ ,  $C_{j+1} = (r_j(I_j), t_j + dt_j)$  with  $r_j, dt_j$  as in [27]:

$$\begin{aligned}
 & - r_j = rule^k, k \in \{1, \dots, m\} \text{ and } k \text{ satisfies } \sum_{i=1}^{k-1} p_j^i < ran'_j \cdot a_j \leq \sum_{i=1}^k p_j^i \\
 & - dt_j = (-1/a_j) \ln(ran''_j)
 \end{aligned}$$

where  $ran'_j, ran''_j$  are two random variables over the sample space  $(0, 1]$ , uniformly distributed.

- for  $j = h$ ,  $a_j = 0$  or  $t_j \geq t_{fin,\Pi}$ .

In other words, an evolution of  $\Pi$  is a sequence of configurations  $\langle C_0, C_1, \dots, C_j, C_{j+1}, \dots, C_h \rangle$ , starting from the initial configuration of  $\Pi$ , where, given the current configuration  $C_j = (I_j, t_j)$ , the next one  $C_{j+1} = (I_{j+1}, t_{j+1})$ , is obtained by applying the rule  $r_j$  to the current instantaneous description  $I_j$  and adding  $dt_j$  to the current time  $t_j$ . The rule  $r_j$  and the associated  $dt_j$  are determined by Gillespie's theory

of chemically reacting systems [15] applied to the current instantaneous description  $I_j$  (i.e., effectively, the rule with the shortest waiting time is selected to be executed). The evolution of the system halts when all rules have probability zero to be executed (following from the fact that  $a_j = 0$ ) or when the current time is greater or equal to the specified final time. The detailed explanation of the application of Gillespie's stochastic simulation algorithm [15] to a membrane system with peripheral proteins can be found in [9] and [28]. Similar implementations of the Gillespie algorithm have been already proposed in different models of membrane systems where the algorithm has been specifically adapted and optimized to run in multiple compartments and, for these reasons, referred to as multi-compartment in [17] and [20].

## 2.2 Statistical Model Checking for Membrane Systems with Peripheral Proteins

A stochastic membrane system with peripheral proteins can capture the essential dynamics of a cellular system. It may be used to address questions concerning the interplay between the biochemical processes present in the various compartments and the proteins associated to the cellular membranes. Sometimes, these questions can be resolved in an analytical manner [29] or by executing the model on a computer, [9, 28, 30]. In this chapter we use this latter approach (for an analytical approach see the review [10]). As defined in Sect. 2.1.2, a single evolution of the system produces an outcome that represents the quantities of the involved entities, floating molecules, peripheral proteins and compartments. However, because of the stochastic applications of the rules, each evolution of the system may lead to (a possibly very large number of) different outcomes.

In what follows, we describe the use of statistical model checking to investigate biological systems, where properties of interest are specified using temporal logic.

### 2.2.1 Temporal Logic as a Query Language

Many useful properties of systems can be expressed in terms of maxima, minima or averages of system variables. With more complex reactive systems, such as biological systems, it is often desirable to investigate properties that comprise sequences of events and events dependent on time. Temporal logic provides a formal means to express these properties and remains reasonably intuitive for moderately complex properties. In this chapter we use temporal logic as a query language to investigate temporal properties of a biological system. We have developed a statistical model checker based on the logic of PLASMA [24]. This logic is also similar to the *bounded linear temporal logic* (BLTL) of [21]. Specifically, we have constructed a tool using PLASMA-lab [31, 32], a statistical model checking library that works with an external simulator. We have thus created a simulator that implements a language of stochastic membrane systems with peripheral and integral proteins [33].



For the purposes of exposition, a logical property  $\phi$  of our model checker is constructed using the following abstract syntax:

$$\begin{aligned}\phi &= \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid F_{\leq b}\phi \mid G_{\leq b}\phi \mid \phi U_{\leq b}\phi \mid X\phi \mid \alpha \\ \alpha &= \text{numeric} (> \mid \geq \mid = \mid \leq \mid <) \text{numeric}\end{aligned}$$

$\vee$ ,  $\wedge$  and  $\neg$  are the standard logical connectives *or*, *and* and *not*.  $\alpha$  is an atomic proposition constructed from numeric expressions of constants and system variables using the standard relational operators.  $X$  is the *next* temporal operator ( $X\phi$  is true iff  $\phi$  is true on the next step).  $F$ ,  $G$  and  $U$  are temporal operators bounded by a closed interval  $[0, b]$ , where  $b$  may refer to steps or time. We use the notation  $\phi_t$  and  $\psi_t$  to denote the value of the propositions  $\phi$  and  $\psi$  at step or time  $t$ .  $F$  is the *finally* or *eventually* operator ( $F_{\leq b}\phi$  is true iff  $\exists t \in [0, b] : \phi_t$  is true).  $G$  is the *globally* or *always* operator ( $G_{\leq b}\phi$  is true iff  $\forall t \in [0, b] : \phi_t$  is true).  $U$  is the *until* operator ( $\psi U_{\leq b}\phi$  is true iff  $\exists t \in [0, b] : \phi_t$  is true  $\wedge (t = 0 \vee \forall t' \in [0, t[: \psi_{t'}$  is true). In the case of nested temporal operators, the time bound of an inner temporal operator is relative to the time bound of its directly enclosing operator. Hence, e.g.,  $F_{\leq 3}G_{\leq 4}\phi$  is true iff  $\exists t \in [0, 3], \forall t' \in [t, t + 4] : \phi_{t'}$  is true.

Statistical model checking works by verifying a property  $\phi$  against  $N \in \mathbb{N}$  independent simulation runs. Each simulation run evaluates to true or false and the probability that  $\phi$  is true on an arbitrary execution of the system is estimated by the standard Monte Carlo estimator  $\frac{1}{N} \sum_{i=1}^N \mathbb{1}(\phi)$ , where  $\mathbb{1}(\cdot)$  is an indicator function that returns one if its argument is true and 0 otherwise. To quantify the confidence of the estimate, in this chapter we use a Chernoff bound [22] that guarantees, for given  $N$ , that the absolute error of the estimate is less than  $\epsilon$  with probability  $\delta$ , where  $2\epsilon N = \ln(2/\delta)$ .

## 2.3 A Case Study: The Role of Estrogen in Cellular Mitosis and DNA Damage

At a cellular level, life is punctuated by the recurrence of four major phases: Gap 1 ( $G1$ ), S, Gap 2 ( $G2$ ), and M.  $G1$  is in-between mitosis and DNA replication and is responsible for cell growth. The transition occurring at the restriction point (called R) during the  $G1$  phase commits a cell to the proliferative cycle. If the conditions that enforce this transition are not present, the cell exits the cell cycle and enters a non-proliferative phase (called  $G0$ ) during which cell growth, segregation and apoptosis occur. Replication of DNA takes place during the synthesis phase (called S). It is followed by a second gap phase responsible for cell growth and preparation for division. Mitosis and production of two daughter cells occur in the M phase. Switches from one phase to another are canonically regulated by a family of Cyclins that act as regulatory subunits for the Cyclin-Dependent Kinases (CDKs). According to the actual phase of the cell cycle, a disparate number of chemicals interact with

the Cyclin-CDKs complexes to prevent or favour their move into the nucleus and, consequently, to block or promote the next phase transition [34].

Here we focus on the pre-mitosis G2 phase and model the contention on the Mitosis Promoting Factor (MPF) (i.e., the Cyclin B1/2-CDKs complex), after the occurrence of DNA damage, by two key contributors: p53, the “guardian of the genome” [35], and the estrogens. The predominance of one’s function over the other results in a proliferative rather than in a quiescent state of the cell.

It is known that p53 is a crucial protein in multicellular organisms, where it regulates the cell cycle and functions as a tumour suppressor. p53 has many mechanisms of anticancer function. It can (i) activate DNA repair proteins when DNA has sustained damage; (ii) induce growth arrest by holding the cell cycle at the G1/S regulation point on DNA damage recognition; (iii) initiate apoptosis, the programmed cell death, if DNA damage proves to be irreparable. In G2 phase and after DNA damage, activated p53 binds DNA and induces expression of 14-3-3- $\sigma$  (a.k.a. Stratifin) [36]. Stratifin mRNA exits the nucleus and, after translation, obstructs cell cycle entry by sequestering MPF, thereby preventing its shuttling to the nucleus [37].

Conversely, estrogens, the primary female sex hormones, promote cell cycle progression. They are intracellular proteins present both on the cell surface membrane and in the cytosol. Their actions are assumed to be mediated by estrogen receptors (ERs) which are found in different ratios in the different tissues of the body:

- ER $_{\alpha}$ : endometrium, breast cancer cells, ovarian stroma cells and hypothalamus.
- ER $_{\beta}$ : kidney, brain, bone, heart, lungs, intestinal mucosa, prostate and endothelial cells.

ERs actions can be selectively enhanced or disabled by some estrogen receptor *modulators*, in accordance with the binding affinity level of each estrogenic compound. In the classic model, the *estrogen 17 beta estradiol* binds to the ER, causing displacement of chaperone proteins. Dimers of the estrogen-ER complexes can then act as transcription factors by binding to specific estrogen response element (ERE) sequences in the promoters of target genes, evoking a wide range of transcriptional responses.

Efp, a RING-finger-dependent ubiquitin ligase, is a relevant target gene product of ER $_{\alpha}$ . It is predominantly expressed in various female organs and is responsible for the proteolysis of 14-3-3- $\sigma$  and then it is essential for estrogen-dependent cell proliferation. Its transcription is mediated by the estrogen-ER $_{\alpha}$  complex which enters the nucleus and binds to its ERE. The Efp mRNA can exit the nucleus, translate and eventually bind to the complex stratifin-MPF, floating in the cytoplasm in an inactive form. The newly formed complex dissociates into a macromolecule of ubiquitinated stratifin and one active MPF complex. While the former is targeted for death by proteolysis, the latter can enter the nucleus and promote mitosis progression.

The described pathway is collected from the Biocarta Pathway Database and redrawn using Systems Biology Graphical Notation (SBGN) glyphs in Fig. 2.6.

The cell cycle pathway is crucial to the understanding of cancer because one of the hallmarks of cancer is the uncontrolled proliferation of abnormal or damaged



There are many synthetic molecules on the market that play the role of antagonist to cancer. Tamoxifen, Raloxifene and Anastrozole are some of the most representative. Their antagonism makes sense when coupled with true agonists, which are molecules that typically bind to receptors of a cell and trigger a response by that cell. Agonists often mimic the action of a naturally occurring substance, whereas antagonists block the action of the agonists or cause an action opposite to that of the agonists. The current accepted definition of receptor antagonist is based on the *receptor occupancy* model: agonists are thought to turn on a single cellular response by binding to the receptor, thus initiating a biochemical mechanism for change within a cell. Antagonists are thought to turn off that response by blocking the receptor from the agonist. Whenever the action of the antagonist results to be irreversible, that is its effect lasts throughout the lifetime of the antagonist itself, its dynamics can be described as a key broken off in the lock that prevents any other key from being inserted.

These antagonists are essentially *prodrugs*, and we have added them, in an appropriate way, to the described pathway (see Fig. 2.6). In particular, Tamoxifen, an estrogen blocker that belongs to the class of non-steroidal anti-estrogens, causes cells to remain in the G0 and G1 phases of the cell cycle. In particular, it fights breast cancer by competing with estrogen for space on the receptors of the tumour tissue. Each molecule of Tamoxifen that binds to a receptor prevents an estrogen molecule from engaging at the same place. This can facilitate the treatment of cancer because without a continuous supply of estrogen, cancer cells do not develop and the ability of the tumour to spread is reduced.

Recent experimental work has recognised that drug therapies may be more effective when linked to specific phases of the cell cycle [38] —so-called *chronopharmaceuticals* used in *chronotherapy* —so we specifically consider the time (delay) of the damage with respect to availability of cyclins (the molecules that control mitosis). In this chapter we propose a preliminary study on the interplay between the time of the DNA damage, the amount of the damage and the presence of antagonists.

## 2.4 Methodology and Results

Using PLASMA-lab [32], we have implemented a statistical model checker that allows us to analyse the evolutions of a stochastic membrane systems with peripheral proteins (defined in Sect. 2.1.2), specified in an appropriate syntax and using queries expressed in temporal logic (defined in Sect. 2.2.1). The integrated computational platform can be found at the web-page [26].

The biological model used in our investigation (discussed in Sect. 2.3) is described in terms of a membrane system in Fig. 2.7, where the corresponding simulator script can also be found. In our study, we explicitly consider as parameters the amount of damage, the delay time of the damage and the amount of antagonist. We model the existence of damage by the production of *p53*, a well known indicator, that results from a damage signal that is instantiated as a quantity of molecules denoted

$\Pi$  system *estr*

$V_{estr} = \{damage, p53, ERalpha\_estrogen, Efp,$   
*stratifin*, *CDKs.Cyclin.B1.2*, *MITOSIS*,  
*estrogen*, *ERalpha*, *CDKs.Cyclin.B1.2.stratifin*,  
*Ub.Ligase*, *CDKs.Cyclin.B1.2.stratifin.ubiquitin*,  
*stratifin.ubiquitin* }

$r_1 : [damage \rightarrow p53]_{||}^{nucleopl}$   
 $r_2 : [ ]_{||}^{nucleopl} ERalpha\_estrogen \rightarrow [ERalpha\_estrogen]_{||}^{nucleopl}$   
 $r_3 : [ERalpha\_estrogen]_{||}^{nucleopl} \rightarrow [ ]_{||}^{nucleopl} ERalpha\_estrogen$   
 $r_4 : [ERalpha\_estrogen \rightarrow ERalpha\_estrogen + Efp]_{||}^{nucleopl}$   
 $r_5 : [Efp]_{||}^{nucleopl} \rightarrow [ ]_{||}^{nucleopl} Efp$   
 $r_6 : [p53 \rightarrow p53 + stratifin]_{||}^{nucleopl}$   
 $r_7 : [stratifin]_{||}^{nucleopl} \rightarrow [ ]_{||}^{nucleopl} stratifin$   
 $r_8 : [ ]_{||}^{nucleopl} CDKs.Cyclin.B1.2 \rightarrow [CDKs.Cyclin.B1.2]_{||}^{nucleopl}$   
 $r_9 : [CDKs.Cyclin.B1.2]_{||}^{nucleopl} \rightarrow [ ]_{||}^{nucleopl} CDKs.Cyclin.B1.2$   
 $r_{10} : [CDKs.Cyclin.B1.2 \rightarrow MITOSIS]_{||}^{nucleopl}$

$r_{11} : [ ]_{||}^{cyto} ERalpha\_estrogen \rightarrow [ ]_{||}^{cyto} ERalpha\_estrogen$   
 $r_{12} : [ ]_{||}^{cyto} ERalpha\_estrogen \rightarrow [ERalpha\_estrogen]_{||}^{cyto}$   
 $r_{13} : [CDKs.Cyclin.B1.2 + stratifin \rightarrow$   
 $CDKs.Cyclin.B1.2.stratifin]_{||}^{cyto}$   
 $r_{14} : [Efp + Ub.Ligase + CDKs.Cyclin.B1.2.stratifin \rightarrow$   
 $CDKs.Cyclin.B1.2.stratifin.ubiquitin + Ub.Ligase]_{||}^{cyto}$   
 $r_{15} : [CDKs.Cyclin.B1.2.stratifin.ubiquitin \rightarrow$   
 $CDKs.Cyclin.B1.2 + stratifin.ubiquitin]_{||}^{cyto}$   
 $r_{16} : [stratifin.ubiquitin \rightarrow \lambda]_{||}^{cyto}$   
 $r_{17} : [ ]_{||}^{cyto} ERalpha\_antagonist \rightarrow [ ]_{||}^{cyto} ERalpha\_antagonist$

$\mu_{estr} = [ ]_{||}^{nucleopl} ]_{||}^{cyto}$

$w_{cyto} = CDKs.Cyclin.B1.2^{1000}, Ub.Ligase^{1000}$   
 $w_{nucleopl} = \lambda$

$w_0 = estrogen^{10000}$   
 $(u_{nucleopl} = \lambda, v_{nucleopl} = \lambda, x_{nucleopl} = \lambda)$   
 $(u_{cyto} = \lambda, v_{cyto} = \lambda, x_{cyto} = ERalpha^{1000})$   
 $t_{in} = 0, t_{fin} = 20.000$   
 $rate(r_3) = 0.3, rate(r_8) = 0.06,$   
 $rate(r_9) = 0.03, rate(r_{11}) = 0.1$

## Simulator script

object *damage*, *p53*, *ERalpha\_estrogen*, *Efp*,  
*stratifin*, *CDKs.Cyclin.B1.2*, *MITOSIS*,  
*estrogen*, *ERalpha*, *CDKs.Cyclin.B1.2.stratifin*,  
*Ub.Ligase*, *CDKs.Cyclin.B1.2.stratifin.ubiquitin*,  
*stratifin.ubiquitin*

rule *nucleoplasm\_rules*  
{  
  *damage*  $\rightarrow$  *p53*  
  || + *ERalpha\_estrogen*  $\rightarrow$  *ERalpha\_estrogen* + ||  
  *ERalpha\_estrogen* + || 0.3  $\rightarrow$  || + *ERalpha\_estrogen*  
  *ERalpha\_estrogen*  $\rightarrow$  *ERalpha\_estrogen* + *Efp*  
  *Efp* + ||  $\rightarrow$  || + *Efp*  
  *p53*  $\rightarrow$  *p53* + *stratifin*  
  *stratifin* + ||  $\rightarrow$  || + *stratifin*  
  || + *CDKs.Cyclin.B1.2* 0.06  $\rightarrow$  *CDKs.Cyclin.B1.2* + ||  
  *CDKs.Cyclin.B1.2* + || 0.03  $\rightarrow$  || + *CDKs.Cyclin.B1.2*  
  *CDKs.Cyclin.B1.2*  $\rightarrow$  *MITOSIS*  
}

rule *cytoplasm\_rules*  
{  
  || *ERalpha* + *estrogen* 0.1  $\rightarrow$  || *ERalpha\_estrogen*  
  || *ERalpha\_estrogen*  $\rightarrow$  *ERalpha\_estrogen* + ||  
  *CDKs.Cyclin.B1.2* + *stratifin*  $\rightarrow$   
  *CDKs.Cyclin.B1.2.stratifin*  
  *Efp* + *Ub.Ligase* + *CDKs.Cyclin.B1.2.stratifin*  $\rightarrow$   
  *CDKs.Cyclin.B1.2.stratifin.ubiquitin* + *Ub.Ligase*  
  *CDKs.Cyclin.B1.2.stratifin.ubiquitin*  $\rightarrow$   
  *CDKs.Cyclin.B1.2* + *stratifin.ubiquitin*  
  *stratifin.ubiquitin*  $\rightarrow$  \*  
  || *ERalpha* + *antagonist*  $\rightarrow$  || *ERalpha\_antagonist*  
}

compartment  
  *nucleoplasm* [1000 *damage*@500, *nucleoplasm\_rules*]  
compartment *cytoplasm* [1000 *CDKs.Cyclin.B1.2*,  
1000 *Ub.Ligase*, *nucleoplasm*,  
*cytoplasm\_rules* : ||1000*ERalpha*]  
system *cytoplasm*, 10000 *estrogen*

evolve 0–20000

**Fig. 2.7** A stochastic membrane systems with peripheral and integral proteins that presents the P53-independent G2/M cell cycle arrest pathway, with data collected from the Biocarta Pathways Database, described in Fig. 2.6 and ranges of proportionality between the coefficients obtained using preliminary western-blotting experiments. On the *left* side we present the formal language model following the formal syntax presented in Sect. 2.1.2 (the membrane labels cyto and nucleopl stand for cytoplasm and nucleoplasm, respectively); on the *right* side the equivalent simulator script using the appropriate syntax. The complete description of the simulator syntax can be found in [9, 28] and at the platform webpage [26]

as *damage*. Our modelling language allows us to inject an amount of damage at a specific time, using the syntax *amount damage@delay*, where “*amount*” is a number of *damage* molecules and “*delay*” is the value of the time delay from the start of

the simulation. As an indicator of mitosis, we define a molecular species denoted *MITOSIS*, that is produced by the cyclins in the nucleus. For the purposes of our investigation, we set an arbitrary minimum of 300 molecules of *MITOSIS* to indicate that mitosis will proceed. Our results remain qualitatively similar if this value is changed.

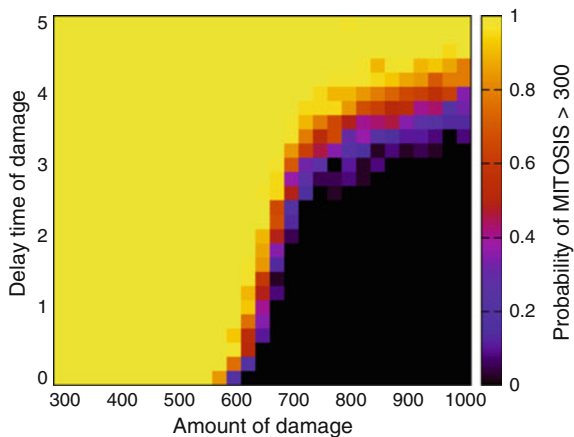
To estimate the probability of mitosis we consider the following temporal logic property:

$$F_{\leq \text{delay}} \text{ damage} = 0 \wedge (X \text{ damage} = \text{amount} \wedge (F_{\leq 20} \text{ MITOSIS} > 300)) \quad (2.1)$$

This property states that in the interval  $[0, \text{delay}]$  there will be a time when  $\text{damage} = 0$  and in the next state  $\text{damage} = \text{amount}$  and within 20 time units  $\text{MITOSIS} > 300$ . In our experiments, the value of  $\text{delay}$  is set to the time at which we inject damage and  $\text{amount}$  to the amount of  $\text{damage}$  injected. Hence, the first part of the property is guaranteed to be satisfied and is used to detect the precise step that damage occurs.<sup>1</sup> This allows the remainder of the property to be timed relative to the damage event. The value of 20 time units is chosen to be sufficiently long to capture all interesting behaviour following the damage.

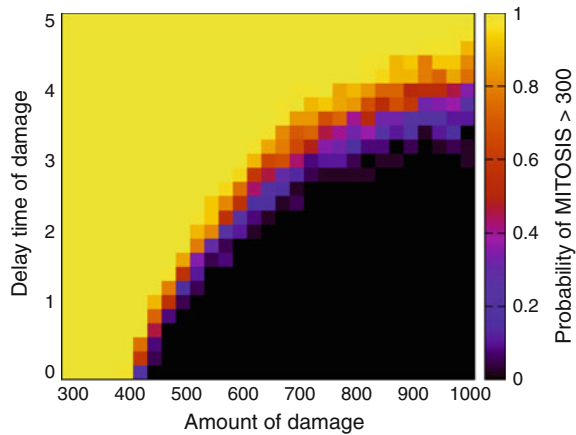
We performed statistical model checking on the model with our model checking tool using the property (2.1). Figure 2.8 illustrates the results of considering amounts of damage in the range  $[300, 1000]$  with increments of 25 and delays in the range  $[0, 5]$  with increments of 0.2. Each point is the result of 37 simulations, which is sufficient (according to the Chernoff bound defined in Sect. 2.2.1) to give a confidence of  $\pm 0.1$  with probability 0.95. 0.95 (95 %) is a standard high level of confidence, while  $\pm 0.1$  is sufficiently precise to resolve the detail in the figure. Figure 2.9 is the result of the same experiments, but with 1,000 *antagonist*. The results for each figure were generated on a single machine (Intel Core i7 2.8 Ghz, 4 GB RAM) in less

**Fig. 2.8** The effect of amount and delay of damage on mitosis: the probability of  $\text{MITOSIS} > 300$  for various amounts and time delay of damage. The figure illustrates the general trend that the probability of mitosis increases with increasing delay and decreasing damage



<sup>1</sup> For zero delay we use the simpler property  $\text{damage} = \text{amount} \wedge (F_{\leq 20} \text{ MITOSIS} > 300)$

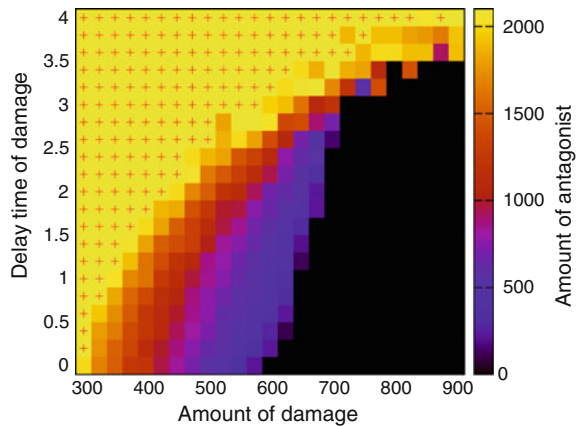
**Fig. 2.9** The effect of antagonist on the response to damage: the probability of  $MITOSIS > 300$  for various amounts and delay of damage in the presence of 1,000 antagonist. In comparison with Fig. 2.8, the figure demonstrates that antagonist increases the sensitivity of the system to damage



than an hour. The general trends are clear in both figures: the probability of mitosis increases with increasing delay and decreases with increasing damage; above a delay of about four time units, mitosis is guaranteed, independent of amount of damage. This confirms our expectation that cell damage causes cell cycle arrest and that if the damage occurs after mitosis has begun, it will be too late to have an effect. Comparing Figs. 2.8 and 2.9, we see that the addition of antagonist increases the region of the figure where the cell cycle is arrested. In particular, the system is more sensitive to lower amounts of damage, confirming our understanding of the effect of an antagonist. We also note that the addition of antagonist has much less effect with respect to delay.

Having observed that the presence of antagonist increases the sensitivity of the system to damage, we investigated the amount of antagonist required to cause cell cycle arrest. The results are plotted in Fig. 2.10 and were achieved in the following way. We estimated the probability of mitosis, as defined by the property (2.1), considering three parameters: amount of damage in the range [300, 900] with increments of 25, delay of damage in the range [0, 4] with increments of 0.2, and amount of antagonist in the range [0, 2100] with increments of 100. Thus, for each combination of amount and delay of damage, we constructed a sequence of probabilities corresponding to the amounts of antagonist, each estimated with the same level of confidence used for Figs. 2.8 and 2.9. The results were generated on a single machine (Intel Core i7 2.8 Ghz, 4 GB RAM) in less than 14 h. Three qualitatively distinct sequences of probabilities emerged: (i) probabilities consistently below 0.5 for all considered values of antagonist (black area in Fig. 2.10); (ii) probabilities consistently above 0.5 for all considered values of antagonist (yellow area marked with ‘+’ symbols) and (iii) probabilities decreasing from values above 0.5 to values below 0.5 with increasing antagonist. Sequences of type (i) correspond to a range of parameters where cell cycle arrest is inevitable, regardless of antagonist. Sequences of type (ii) correspond to a range of parameters where either mitosis is inevitable or more than the maximum amount of antagonist that we tried is required to cause cell cycle arrest. From

**Fig. 2.10** The required dosage of antagonist to prevent mitosis: the amount of antagonist required to make the probability of *MITOSIS* > 300 less than 0.5 when previously it was greater than 0.5. In the black areas the probability of *MITOSIS* > 300 is less than 0.5, independent of antagonist. The ‘+’ symbol indicates that at least 2,100 antagonist would be required to prevent mitosis



sequences of type (iii) we were able to estimate the amount of antagonist corresponding to the transition value of 0.5, by interpolating between two adjacent points or from a single point if, by chance, its value was exactly 0.5.

## 2.5 Conclusions

Even though the life cycle of cells is not generally synchronized, we recall here that replication is subject to the presence of cyclins, that are released in a timely way. Our results then show that it is possible to target cells *on time*, that is, to occupy estrogen receptors (ERs) at time points that are maximally effective. This would convey a double advantage: minimizing the necessary quantity of antagonist and making its effect optimal. The definition of the most effective dosage curve is indeed not a simple task, especially in cases where antagonists cannot be specific for particular cells. For example, in the case of ER+ breast cancer, Tamoxifen is currently taken once or twice a day and it is usually prescribed at 20mg for 5 years. This dosage is due to the unavoidable ineffectiveness of the drug when reaching cells in unfavorable time points, as well as to the fact that Tamoxifen does not specifically target breast cancer cells; its molecules circulate within the body and target any cell that contains an available ER. The consequence of this is that while Tamoxifen works as an anti-estrogen for the breast, it acts as estrogen (i.e., agonist) in the uterus and, to a lesser extent, in the heart, blood vessels and bones. In cases like this, a tuned *chronotherapy* cannot eliminate the risk of side effects, but can drastically reduce it. The presented results show that the use of statistical model checking in membrane systems could be helpful to individuate the appropriate time-dependent dosage of antagonists in cancer treatments [39].



## References

1. Gh. Păun, G. Rozenberg, A. Salomaa, *The Oxford Handbook of Membrane Computing* (Oxford University Press Inc., New York, 2010)
2. M. Pérez-Jiménez, F. Romero-Campero, P systems, a new computational modelling tool for systems biology. *T. Comp. Sys. Biol.* **4220**, 176–197 (2006)
3. Gh Păun, Computing with membranes. *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000)
4. L. Cardelli, Brane calculi, in *Computational Methods in Systems Biology*, ed. by V. Danos, V. Schachter. *Lecture Notes in Computer Science*, vol. 3082 (Springer, Berlin, 2005), pp. 257–278
5. L. Cardelli, Gh Păun, An universality result for a (mem)brane calculus based on mate/drip operations. *Int. J. Found. Comput. Sci.* **17**(1), 49–68 (2006)
6. R. Brijder, M. Cavaliere, A. Riscos-Núñez, G. Rozenberg, and D. Sburlan. Membrane systems with marked membranes. *Electronic Notes in Theoretical Computer Science*, **171**(2), 25–36, (2007), in *Proceedings of the First Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2006)*.
7. M. Cavaliere, S. Sedwards, Membrane systems with peripheral proteins: transport and evolution. *Electronic Notes in Theoretical Computer Science*, **171**(2), 37–53 (2007). *Proceedings of the First Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2006)*
8. D. Bray, A. Johnson, J. Lewis, M. Raff, K. Robert, P. Walter, B. Alberts, *Essential Cell Biology: An Introduction to the Molecular Biology of the Cell*, vol. 1 (Garland Publishing, Inc., New York, 1998)
9. M. Cavaliere, S. Sedwards, Modelling Cellular Processes Using Membrane Systems with Peripheral and Integral Proteins, in *Proceedings of the 2006 International Conference on Computational Methods in Systems Biology, CMSB'06*. (Springer, Berlin, 2006) pp. 108–126
10. M. Cavaliere, S.N. Krishna, Gh. Păun, A. Păun, P systems with objects on membranes, in *The Oxford Handbook of Membrane Computing*, ed. by Gh. Păun, G. Rozenberg, A. Salomaa. (Oxford University Press, New York, 2010) pp. 363–388
11. A. Păun, M. Păun, A. Rodríguez-Patón, M. Sidoroff, P systems with proteins on membranes: a survey. *Int. J. Found. Comput. Sci.* **22**(1), 39–53 (2011)
12. J. Fisher, Th.A. Henzinger, Executable cell biology. *Nat. Biotechnol.* **25**, 1239–1249 (2007)
13. E.M. Clarke, E.A. Emerson, J. Sifakis, Model checking: algorithmic verification and debugging. *Commun. ACM* **52**(11), 74–84 (2009)
14. E.M. Clarke, O. Grumberg, D.A. Peled, *Model Checking* (The MIT Press, Cambridge, 1999)
15. D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977)
16. C. Baier, J.-P. Katoen, *Principles of Model Checking* (MIT Press, Cambridge, 2008)
17. F.J. Romero-Campero, M. Gheorghe, L. Bianco, D. Pescini, M.J. Pérez-Jiménez, R. Ceterchi, Towards probabilistic model checking on P systems using PRISM, in *Proceedings of the 7th International Conference on Membrane Computing, WMC'06*. (Springer, Berlin, 2006) pp. 477–495
18. J. Heath et al., Probabilistic model checking of complex biological pathways. *Theor. Comput. Sci.* **391**, 239–257 (2008)
19. M. Kwiatkowska, G. Norman, D. Parker, Using probabilistic model checking in systems biology. *SIGMETRICS Perform. Eval. Rev.* **35**(4), 14–21 (2008)
20. J. Blakes, J. Twycross, F.J. Romero-Campero, N. Krasnogor, The infobiotics workbench: an integrated in silico modelling platform for systems and synthetic biology. *Bioinformatics* **27**(23), 3323–3324 (2011)
21. E.M. Clarke, J.R. Faeder, C.J. Langmead, L.A. Harris, S. Jha, A. Legay, Statistical model checking in biolab: applications to the automated analysis of t-cell receptor signaling pathway, in ed. by M. Heiner, A.M. Uhrmacher *Computational Methods in Systems Biology*, *Lecture Notes in Computer Science*, vol. 5307 (Springer, Berlin, 2008), pp. 231–250
22. H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.* **23**(4), 493–507 (1952)

23. The PRISM website. [www.prismmodelchecker.org](http://www.prismmodelchecker.org)
24. C. Jegourel, A. Legay, and S. Sedwards. A platform for high performance statistical model checking: plasma, in *Tools and Algorithms for the Construction and Analysis of Systems*, ed. by C. Flanagan, B. König. Lecture Notes in Computer Science, vol. 7214 (Springer, Berlin, 2012) pp. 498–503. doi: 10.1007/978-3-642-28756-5\_37
25. J.E. Hopcroft, R. Motwani, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd edn. (Addison-Wesley Longman Publishing Co. Inc., Boston, 2006)
26. Platform webpage. <https://sites.google.com/site/cytosim/smc>
27. D.T. Gillespie, Stochastic simulation of chemical kinetics. *Ann. Rev. Phys. Chem.* **58**(1), 35–55 (2007)
28. S. Sedwards, T. Mazza, Cyto-sim: a formal language model and stochastic simulator of membrane-enclosed biochemical processes. *Bioinformatics* **23**(20), 2800–2802 (2007)
29. M. Cavaliere, S. Sedwards, Decision problems in membrane systems with peripheral proteins, transport and evolution. *Theor. Comput. Sci.* **404**(1–2), 40–51 (2008)
30. M. Cavaliere, T. Mazza, A (natural) computing perspective on cellular processes, in *Elements of Computational Systems Biology* ed. by H.M. Lodhi, S.H. Muggleton. (Wiley, 2010) pp. 115–138
31. B. Boyer, K. Corre, A. Legay, S. Sedwards, PLASMA-lab : a flexible, distributable statistical model checking library, in *Quantitative Evaluation of Systems* 2013
32. PLASMA-lab project webpage. <https://project.inria.fr/plasma-lab>
33. Cyto-Sim language webpage. <https://sites.google.com/site/cytosim/language>
34. P. Ballarini, T. Mazza, A. Palmisano, A. Csikasz-Nagy, Studying irreversible transitions in a model of cell cycle regulation. *Electron. Notes Theor. Comput. Sci.* **232**, 39–53 (2009)
35. D.P. Lane, p53, guardian of the genome. *Nature* **358**(6381), 15–16 (1992)
36. H. Hermeking, C. Lengauer, K. Polyak, T.C. He, L. Zhang, S. Thiagalingam, K.W. Kinzler, B. Vogelstein, 14–3–3 sigma is a p53-regulated inhibitor of g2/m progression. *Mol. Cell.* **1**(1), 3–11 (1997)
37. C. Laronga, H.-Y. Yang, C. Neal, M.-H. Lee, Association of the cyclin-dependent kinases and 14–3–3 sigma negatively regulates cell cycle progression. *J. Biol. Chem.* **275**(30), 23106–23112 (2000)
38. K. Ross, Circadian rhythms play role in cancer research. *J. Natl. Cancer Inst.* **98**(12), 806–807 (2006)
39. H. Kitano, A robustness-based approach to systems-oriented drug design. *Nat. Rev. Drug Discov.* **5**, 202–210 (2007)

# Chapter 3

## Molecular Diffusion and Compartmentalization in Signal Transduction Pathways: An Application of Membrane Systems to the Study of Bacterial Chemotaxis

Paolo Cazzaniga, Daniela Besozzi, Dario Pescini and Giancarlo Mauri

**Abstract** In this chapter we present an application of membrane systems to the study of intracellular diffusive processes. In particular, a class of membrane systems, called  $\tau$ -DPP, is used for the modeling, simulation and analysis of bacterial chemotaxis. Two different models of this signal transduction pathway are presented. The first is a single volume model used to investigate the properties of bacterial chemotaxis and to analyze the effects of different perturbations (deletion of chemotactic proteins, addition of distinct amounts of external ligand, effect of different methylation states of the receptors) on the system dynamics. The second model represents a multivolume extension of the former, and it is exploited for the analysis of the diffusive processes that give rise to the formation of concentration gradients throughout the bacterial cytoplasm. The outcome of stochastic simulations of both models are exploited to analyze the process of synchronization of flagella, in order to evaluate the running and tumbling time intervals of bacterial cells.

---

P. Cazzaniga (✉)

Dipartimento di Scienze Umane e Sociali, Università degli Studi di Bergamo,  
Piazzale S. Agostino 2, 24129 Bergamo, Italy  
e-mail: paolo.cazzaniga@unibg.it

D. Besozzi

Dipartimento di Informatica, Università degli Studi di Milano, Via Comelico 39,  
20135 Milano, Italy  
e-mail: besozzi@di.unimi.it

D. Pescini

Dipartimento di Statistica e Metodi Quantitativi, Università degli Studi di Milano-Bicocca,  
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy  
e-mail: dario.pescini@unimib.it

G. Mauri

Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di  
Milano-Bicocca, Viale Sarca 336, 20126 Milano, Italy  
e-mail: mauri@disco.unimib.it

### 3.1 Introduction

The development of methodologies for mathematical modeling and computer simulations is nowadays allowing to study the complexity of many biological systems. In the last decades, a wide variety of models of cellular processes based on different formalisms have been proposed. Recently, by taking advantage of the biologically inspired aspects of their underlying structure, membrane systems (or P systems) have been exploited for the modeling of biological systems and for the investigation of their dynamical properties. Many classes of P systems have been defined by taking inspiration from various aspects of living cells, and several applications have reported the potential of P systems in this field of research [1, 2]. For a comprehensive overview of basic P systems, of other classes lately introduced and their application in Computer Science and Biology, we refer the interested reader to [3], and to the P systems Web Page [4].

Here, we focus on an application of P systems for the modeling of diffusive events in a signal transduction pathway, namely, bacterial chemotaxis. Chemotaxis is an efficient signal transduction pathway which allows bacterial cells to move directionally, in response to specific attractants or repellents occurring in their surroundings. The pathway consists of several transmembrane and cytoplasmic proteins acting as signal sensors and response regulators [5], which rule the reversal of the flagellar motor (governed by the phosphorylation and dephosphorylation of a key protein, CheY). In homogeneous environments, this process induces a switch between running and tumbling movements, with a frequency that allows a temporal sampling (through random walks) of the surrounding space. On the contrary, in the presence of a gradient of attractants or repellents, bacteria are able to respond quickly by reducing the frequency of flagellar reversal between clockwise and counterclockwise rotations, which cause a longer running motion in a biased direction. The frequency of switching is then reset to the random walk level if the concentration of the external ligands remains constant in time. At the molecular scale, this adaptation property is implemented by the coordinated action of methyltransferase and methylesterase proteins acting on the transmembrane receptors.

The genetic regulation and biochemical functions of the proteins involved in chemotaxis are well known, and several models have already been proposed to study their complex interplay as well as the robustness of this system [6–11]. In the models we present here, we consider detailed protein-protein interactions for the chemotactic pathway in *E. coli*, in response to attractant molecules, which sum up to 32 molecular species and 62 biochemical reactions. The temporal evolution of the phosphorylated form of CheY (CheYp) is investigated under different conditions, such as the deletion of other proteins involved in the pathway, the addition of distinct amounts of external ligand, and the effect of different methylation states. In particular, we show how to exploit the concept of virtual volumes in membrane systems to describe and simulate the occurrence of intracellular concentration gradients of CheYp, and present the effects of different diffusion coefficients and of a varying number of virtual volumes over the response of this pathway.

Indeed, the requirement that living cells are constituted by diluted and homogeneous compartments can be assumed in some particular cases, but there are several cellular processes in which the effects of spatial heterogeneity—due to diffusive events—must be explicitly considered in order to capture the correct system behavior [12]. In this context, reaction-diffusion (RD) systems are usually used to describe chemical systems where the spatial distribution of chemicals can influence the overall system dynamics. The standard computational approach for RD systems exploits a continuous time and space domain description, based on a set of partial differential equations that can be solved analytically or numerically; in addition, other computational frameworks (e.g., molecular dynamics, Brownian motion, cellular automata) can be used to analyze such kind of systems [13]. Lastly, spatial approaches based on Gillespie’s method [14] were also introduced, so as to represent RD systems as a set of well-stirred chemical reactors that communicate particles with each other (see [12] for more details). These methods, based on a master equation approach, can be more appropriate when biological noise plays a major role on the system dynamics [15, 16]. This formulation adopts a mechanistic perspective on the chemical system, by describing it as a sequence of collision events among molecules. Each collision can lead either to a new chemical compound (reactive collision) or to an elastic scattering (diffusive collision), which does not alter the nature of chemical species but only their distribution in space. The resulting dynamics is the superposition of a typical Brownian motion (random walk) with an interaction/reaction process. In the case that the reaction volume is homogeneous, this picture corresponds to a well-stirred reactor and the dynamics can be tracked by means of a stochastic simulation algorithm [14].

In this context, we exploit the multivolume stochastic simulation algorithm  $\tau$ -DPP [17] to model and simulate diffusive events in bacterial chemotaxis. The outcome of the stochastic simulations of this model are fundamental for the investigation of this system: we exploit these results to analyze the interplay between stochastic fluctuations of CheYp and the number of cellular flagella, which occur in a few units in the individual bacterium (around half a dozen in *E. coli*). The aim of this analysis is to devise the mean time periods during which the cell either performs a running or a tumbling motion, considering both the coordination of flagella and the randomness that is intrinsic in the chemotactic pathway.

Experimental observations show that the running motion requires all flagella to be simultaneously synchronized in the counterclockwise rotation, which occurs when CheYp is not interacting with the proteins regulating the flagellar motor. When at least one flagellum is not coordinated with the others, then the bacterium performs a tumbling movement. To distinguish between these two states, we assume that the cell is sensitive to a threshold level of CheYp, that is evaluated as the mean value of CheYp at steady state. Because of stochastic fluctuations, the amount of CheYp randomly switches from below to above this value, thus reversing the rotation of each flagellum from counterclockwise to clockwise. Therefore, we exploit stochastic simulations to link the synchronization of all flagella to the fluctuations of CheYp as the core component that stands at the basis of chemotactic motions. To this aim, we define a procedure to identify the synchronization of rotations of all flagella, and

we use it to compare the mean time intervals of running and tumbling motions—as well as of the adaptation times after ligand addition—according to a varying number of flagella. It is worth noting that, in this context, the application of deterministic methods (e.g., based on ordinary differential equations) is inappropriate, since the outcome of deterministic simulations would yield a *flat* temporal evolution of CheYp amount, not showing any switch between counterclockwise and clockwise rotations of the flagellar motors. As a consequence, a deterministic model of bacterial chemotaxis would not allow to investigate any biologically interesting aspect related to the synchronization of flagella, that are actually influenced by the stochastic fluctuations of CheYp.

The chapter is structured as follows. In Sect. 3.2 we introduce the basic notions of membrane systems and of  $\tau$ -DPP class. In Sect. 3.3 we present the bacterial chemotaxis system, and the single and multivolume models that we have formalized by means of  $\tau$ -DPP. In Sect. 3.4 we present the results concerning the simulation and analysis of the two models. Section 3.5 concludes the chapter with some final remarks.

## 3.2 A Multivolume Modeling Approach with Membrane Systems

In this section we recall the basic notions of membrane systems and of  $\tau$ -DPP, a computational framework that can be used to formally describe and simulate stochastic models of complex biological systems.

### 3.2.1 Membrane Systems

Membrane systems, also called P systems, were introduced in [18] as a class of unconventional computing devices of distributed, parallel and nondeterministic type, inspired by the compartmental structure and the functioning of living cells. A basic P system is defined by a membrane structure where multisets of objects evolve according to given evolution rules, which also determine the communication of objects between membranes. A *membrane structure* consists of a set of membranes hierarchically embedded in a unique membrane, called the skin membrane. The membrane structure is represented by a string of correctly matching square parentheses, placed in a unique pair of matching parentheses. Each pair of matching parentheses corresponds to a membrane and, usually, membranes are univocally labeled with distinct numbers. For instance, the string  $\mu = [{}_0 [{}_1 ]_1 [{}_2 [{}_3 ]_3 [{}_4 ]_4 ]_2 ]_0$  corresponds to a membrane structure consisting of 5 membranes placed at three hierarchical levels. Moreover, the same membrane structure can be also represented by the string  $\mu' = [{}_0 [{}_2 [{}_4 ]_4 [{}_3 ]_3 ]_2 [{}_1 ]_1 ]_0$ , that is, any pair of matching parentheses at the same hierarchical level can be interchanged, together with their contents; this means that the order of pairs of parentheses is irrelevant, what matters is their respective rela-

tionship. Each membrane identifies a *region*, delimited by it and by the membranes (if any) immediately inside it. The number of membranes in a membrane structure is called the *degree* of the P system. The whole space outside the skin membrane is called the *environment*.

An *object* can be a symbol or a string over a specified finite alphabet  $V$ ; multisets of objects are usually considered in order to describe the presence of multiple copies of any given object inside a membrane. A multiset associated with membrane  $i$  is a map  $M_i : V \rightarrow \mathbf{N}$  which associates a multiplicity to each object in the multiset itself. Inside any membrane  $i$ , objects in  $M_i$  are modified by means of *evolution rules*, which are multiset rewriting rules of the form  $r_i : u \rightarrow v$ , where  $u$  and  $v$  are multisets of objects. The objects from  $v$  have associated target indications which determine the regions where they are to be placed (communicated) after the application of the rule: if  $tar = here$ , then the object remains in the same region; if  $tar = out$ , then the object exits from the region where it is placed and enters the outer region (or even exits the system, if the rule is applied in the skin membrane); if  $tar = in_j$ , then the object enters the membrane labeled with  $j$ ,  $j \neq i$ , assumed it is placed immediately inside the region where the rule is applied (otherwise the rule cannot be applied).

When considering P systems as computing devices, a computation is obtained starting from an initial configuration (described by a fixed membrane structure containing a certain number of objects and rules) and letting the system evolve. A universal clock is assumed to exist: at each step, all rules in all regions are simultaneously applied to all objects which can be the subject of an evolution rule. That is, we say that rules are applied in a maximal parallel manner; as well, membranes evolve simultaneously. If no further rule can be applied, the computation halts and its result is read in a prescribed way.

In what follows, we consider P systems as a modeling tool for biological systems, and not from a computing perspective. To this purpose, membranes and regions are used to characterize cellular compartments or volumes, objects correspond to biochemical species, and rules describe molecular interactions between species. In this context, given a rule  $u \rightarrow v$  and objects  $a_1, \dots, a_k, b_1, \dots, b_h \in V$ , we use the notation  $n_1 a_1 + \dots + n_k a_k \rightarrow m_1 b_1 + \dots + m_h b_h$  to denote the elements in the multiset  $u$  and  $v$ , instead of the classical notation  $u = a_1^{n_1}, \dots, a_k^{n_k}$  and  $v = b_1^{m_1}, \dots, b_h^{m_h}$ . We also talk about *evolution* of the P system instead of *computation*.

### 3.2.2 $\tau$ -DPP

The class of P systems called  $\tau$ -DPP [17] represents an integration of a membrane structure (as defined in Sect. 3.2.1) with the tau-leaping algorithm [19]. *Tau-leaping* is an approximated and faster version of the seminal Gillespie's stochastic simulation algorithm (SSA) [14]. Both algorithms allow to generate the temporal evolution of chemicals contained inside a well stirred volume, in given and fixed experimental conditions. Chemicals interact with each other by means of given reactions, whose physical and chemical properties are encompassed in a specified



stochastic constant associated to each reaction. Reactions are applied according to a probability distribution, that is determined—at each computation step—by the current state of the system (given by the number of molecules of each chemical species) and by the value of all reaction constants. SSA and tau-leaping share the characteristic that repeated (independent) executions will produce different temporal dynamics, even starting from the same initial configuration, thus reflecting the inherent noise of the system. The two algorithms differ with respect to the way reactions are applied. In SSA, only *one* reaction can be applied at each step; the reaction that will be actually simulated, and the waiting time before the next reaction will take place, depend on two independent random numbers drawn from the uniform unit interval  $[0, 1]$ . In tau-leaping, instead, *several* reactions can be chosen and executed simultaneously, by the sampling of Poissonian distributions and by choosing an opportune time increment (we refer to [19] for further details). So doing, the computational burden typical of SSA in simulating large systems, consisting of many reactions and many chemical species, can be largely reduced. Indeed, with tau-leaping it is possible to handle detailed descriptions of many molecular interactions and chemical modifications (e.g., phosphorylation or methylation states of proteins), providing fast and reliable stochastic simulations of mechanistic models of complex biological systems [20]. On the other side, tau-leaping is not guaranteed to reproduce the exact behavior of the system, but the accuracy of the simulation can be controlled.

More precisely, tau-leaping works as follows. Let us denote by  $X$  a well stirred system in thermal equilibrium, consisting of  $S$  molecular species  $s_1, \dots, s_S$ , which can interact through  $R$  chemical reactions  $r_1, \dots, r_R$ . Let  $X_i(t)$  be the number of molecules of chemical  $s_i$  at time  $t$ , and  $\mathbf{x} = \mathbf{X}(t) \equiv (X_1(t), \dots, X_S(t))$  the state of the system at time  $t$ . The aim of the procedure is to fire several reactions for each time interval  $[t, t + \tau)$ . In order to find out which reactions will be executed, we have to calculate the probability that a reaction  $r_j$  will occur in the next infinitesimal time interval  $[t, t + dt)$ , starting from the system state  $\mathbf{x}$ . This probability is given by  $a_j(\mathbf{x})dt$ , which is the *propensity function* of reaction  $r_j$  and is defined as  $a_j(\mathbf{x}) = h_j(\mathbf{x}) \cdot c_j$ , where  $h_j(\mathbf{x})$  is the number of distinct reactant molecules combinations in  $r_j$ , and  $c_j$  is the stochastic constant associated to  $r_j$ . Given a state  $\mathbf{x}$  of the system  $X$ , we denote by  $K_j(\tau, \mathbf{x}, t)$  the exact number of times that a reaction  $r_j$  will be fired in the time interval  $[t, t + \tau)$ , so that  $\mathbf{K}(\tau, \mathbf{x}, t)$  is the exact probability distribution vector (having  $K_j(\tau, \mathbf{x}, t)$  as elements). For arbitrary values of  $\tau$ , calculating the values  $K_j(\tau, \mathbf{x}, t)$  is as difficult as resolving the Chemical Master Equation corresponding to system  $X$ . On the contrary, if  $\tau$  is small enough so that the change in the state  $\mathbf{x}$  during  $[t, t + \tau)$  is so slight that no propensity function will suffer an appreciable change in its value (this is called the *leap condition*), then it is possible to evaluate a good approximation of  $K_j(\tau, \mathbf{x}, t)$  by using the Poisson random variables with mean and variance  $a_j(\mathbf{x}) \cdot \tau$ . Hence, starting from the state  $\mathbf{x}$  and choosing a  $\tau$  value that satisfies the leap condition, the state of the system is updated at time  $t + \tau$  according to  $\mathbf{X}(t + \tau) = \mathbf{x} + \sum_{j=1, \dots, R} \mathbf{v}_j P_j(a_j(\mathbf{x}), \tau)$ , where  $P_j(a_j(\mathbf{x}), \tau)$  denotes an independent sample of the Poisson random variable with mean and variance  $a_j(\mathbf{x}) \cdot \tau$ , and  $\mathbf{v}_j \equiv (v_{1j}, \dots, v_{Sj})$  is the state change vector whose element  $v_{ij}$  represents the stoichiometric change of species  $s_i$  due to reaction  $r_j$ . Summarizing,



each iterative step of the algorithm consists of five stages: (1) generate the maximum changes of each species that satisfy the leap condition; (2) compute the mean and variance of the changes of the propensity functions; (3) evaluate the leap value  $\tau$  exploiting the auxiliary quantities previously computed; (4) toss the reactions to apply; (5) update the system state (see [19] for further details).

The introduction of  $\tau$ -DPP [17] for the investigation of biological systems was motivated by two main problems. The first consists in the fact that the tau-leaping algorithm is only applicable to well stirred chemical reaction systems contained inside a *single* fixed volume. The second problem concerns the improvement of another class of P systems, called DPPs [21], which are membrane systems where probabilities are associated with the rules, and such values vary during the evolution of the system, according to a prescribed strategy. However, DPPs only allow *qualitative* simulations of the system dynamics, since rules are applied according to a universal clock in a parallel manner and no time length is associated to their application.

A solution to the first problem consists in exploiting the framework of DPPs, since the membrane structure is suitable to represent systems consisting of *many* regions [21–23]. The second problem can be solved by extending tau-leaping to the modeling framework provided by DPPs. Within the framework of P systems, this algorithm represents a novel tool for the modeling of multivolume systems able to provide a *quantitative* description of the system dynamics. Briefly, in  $\tau$ -DPP instead of assuming a global clock for the instantaneous and parallel application of the rules, as it is done in DPPs, each step has a different length, computed according to the current system state; then, during the time step, a certain number of rules is selected and executed, according to the tau-leaping algorithm.

$\tau$ -DPP is a computational method which can be used to describe and perform stochastic simulations of complex biological systems. For instance, cellular pathways involving several spatial compartments (as the extracellular ambient, the cytoplasm, the nucleus, etc.), or multicellular systems like bacterial colonies, or multi-patched ecological systems as metapopulations, are all examples of complex systems that could be investigated with  $\tau$ -DPP. Since  $\tau$ -DPP represents a general simulation framework for a broad range of complex systems, in the following we use the generic terms *volume* (or region), *object* (or species) and *rule* (or reaction), to denote the compartment where the molecular species can be modified in some way by a biochemical reaction. The correct behavior of the whole system is achieved by letting all volumes evolve in parallel, and by using the following strategy for the choice of time increments. At each iteration,  $\tau$ -DPP considers the current state of each volume (determined by the current number of objects), and then calculates a time increment independently in each volume (according to the standard tau-leaping algorithm). Then, the smallest time increment is selected and used to evaluate the next-step evolution of the entire system. Since all volumes *locally* evolve according to the same time increment,  $\tau$ -DPP is able to correctly work out the *global* dynamics of the system. Moreover, by adopting this procedure, the simulated evolutions of all volumes get naturally *synchronized* at the end of each iterative step. The synchronization is also necessary—and exploited together with a parallel update of all volumes—

to manage the communication of objects among volumes, whenever prescribed by specific rules.

Formally, a  $\tau$ -DPP  $\Upsilon$  is defined as

$$\Upsilon = (V_1, \dots, V_N, \mu, \mathcal{S}, M_1, \dots, M_N, R_1, \dots, R_N, C_1, \dots, C_N),$$

where:

- $V_1, \dots, V_N, N \geq 1$ , are the volumes of the system;
- $\mu$  is a membrane structure representing the topological arrangement of the volumes;
- $\mathcal{S} = \{s_1, \dots, s_S\}, S \geq 1$ , is the set of objects, that is, the alphabet of the system;
- $M_1, \dots, M_N$  are the multisets over  $\mathcal{S}$  occurring inside the volumes  $V_1, \dots, V_N$ , respectively, representing the internal state of the volumes;
- $R_1, \dots, R_N$  are the sets of rules defined in volumes  $V_1, \dots, V_N$ , respectively. A rule can be of internal or of communication type (as described below);
- $C_1, \dots, C_N$  are the sets of stochastic constants associated to the rules defined in volumes  $V_1, \dots, V_N$ , respectively.

The system  $\Upsilon$  is defined by means of a set of  $N$  volumes organized according to the hierarchy specified by the membrane structure  $\mu$ . The state of the whole system is characterized by all multisets  $M_i$  occurring inside each volume  $V_i$  ( $1 \leq i \leq N$ ).

Inside the volumes, the sets of rules  $R_1, \dots, R_N$  are defined along with the sets of stochastic constants  $C_1, \dots, C_N$ . The stochastic constants are needed, together with a combinatorial function depending on the left-hand side of the rule (as explained in [14]), to compute the probabilities of the rule applications (i.e., their propensity functions).

Each volume  $V_i$  can contain two different kinds of rules, termed *internal* and *communication* rules. An internal rule describes the modification, or evolution of the objects inside the single volume where it is applied, while a communication rule sends the objects from the volume where it is applied to an adjacent volume (possibly modifying the form of these objects during the communication step).

More precisely, internal rules have the general form  $\alpha_1 s_1 + \alpha_2 s_2 + \dots + \alpha_S s_S \rightarrow \beta_1 s_1 + \beta_2 s_2 + \dots + \beta_S s_S$ , where  $s_1, \dots, s_S \in \mathcal{S}$  are distinct object types and  $\alpha_1, \dots, \alpha_S, \beta_1, \dots, \beta_S \in \mathbb{N}$ . For instance,  $s_1, \dots, s_S$  can correspond to molecular species, and, in this case,  $\alpha_1, \dots, \alpha_S, \beta_1, \dots, \beta_S$  represent stoichiometric coefficients. The objects appearing in the left-hand side of the rule are called *reagents*, while the objects on the right-hand side are called *products*. Note that, usually, we consider the case where (at most) two objects appear in the reagents group. The rationale behind this is that we require biochemical reactions to be (at most) of the second-order, since the simultaneous collision and chemical interaction of more than two molecules at a time, has a probability to occur close to zero in real biochemical systems. Moreover, the interaction among more than two molecules can be modeled by using a set of successive reactions with lower order.

When dealing with communication rules inside a volume, besides defining the sets of reagents and products, it is necessary to specify the target volume where

the products of this rule will be sent (this definition can be easily extended in order to assign a different target volume to each object appearing in the set of products). Formally, a communication rule has the form  $\alpha_1 s_1 + \alpha_2 s_2 + \dots + \alpha_S s_S \rightarrow (\beta_1 s_1 + \beta_2 s_2 + \dots + \beta_{SS} s_S, tar)$ , where  $s_1, \dots, s_S \in \mathcal{S}$  are distinct object types,  $\alpha_1, \dots, \alpha_S, \beta_1, \dots, \beta_S \in \mathbb{N}$ , and  $tar$  can be equal to:

- *out*: this means that the products of the rule are “sent outside” the source volume (that is, the region where the rule is applied), to the adjacent outer volume;
- *in<sub>label</sub>*: this means that the products of the rule are “sent inside” the volume with the label specified by the target. These rules are only allowed if the target volume is placed inside the source volume, and the two volumes are adjacent (that is, there is no other volume placed between the source and the target volume).
- *to<sub>label</sub>*: this means that the products of the rule are “sent to” the volume with the label specified by the target. These rules are only allowed if the target volume is adherent to the source volume (that is, there is no other volume placed between the source and the target volume).

The condition that at most two objects appear as reagents is usually required also for communication rules. Communication rules are considered special rules for what concerns the time increment ( $\tau$ ) selection procedure, applied in the first stage of the  $\tau$ -DPP algorithm. For internal rules, indeed, this procedure is exactly the same as the tau-leaping algorithm, where the length of the step is computed by bounding the variation of the molecular amounts. Namely, in order to correctly evaluate the simulation time increment and to describe the behavior of the system with a good approximation, the “largest” value of  $\tau$  that also satisfies the leap condition is chosen, as fully described in [19].

On the contrary, the variation due to communication rules implies a change in the amounts of objects inside *two* different volumes: the reagents inside the source volume, and the products sent to the destination volume. To correctly estimate the value of  $\tau$  when dealing with communication rules, instead of limiting the variation of both reagents and products (as it is done for internal rules), only the variation of the reagents inside the source volume is considered (that is, only the left-hand side of the communication rule is used). Indeed, the value of  $\tau$  is independent from any objects that has to be communicated, since these products will be received in the target volume only at the end of the iteration step. For this reason, for the  $\tau$  selection procedure, the right-hand side of a communication rule is neither considered in the source, nor in the target volume. Obviously, the communicated objects will contribute to update the system state, which takes place at the end of the iteration step, and will be therefore considered to determine the state of the target volume for the next iteration step.

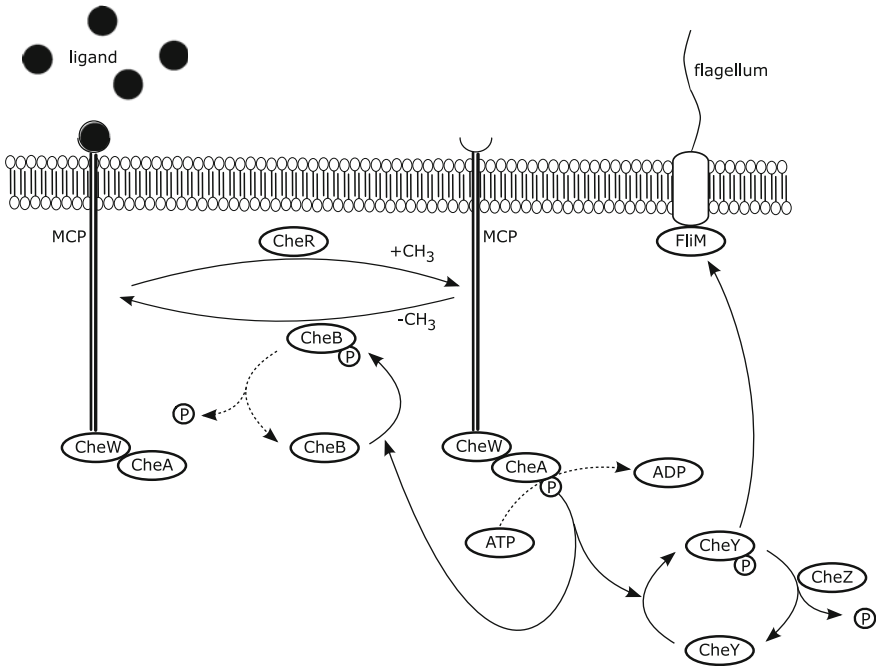
### 3.3 The Modeling of Bacterial Chemotaxis

In this section we introduce the chemotaxis signaling pathway, we define the single volume mechanistic model that describes the molecular interactions therein involved, and then introduce the multivolume model, both formalized by means of  $\tau$ -DPP.

### 3.3.1 Bacterial Chemotaxis

Chemotaxis is a signal transduction pathway that allows swimming bacteria to perform biased movements in ever-changing environments, by efficiently sensing concentration gradients of beneficial or toxic chemicals in their immediate proximity. The binding of ligand molecules triggers an events cascade involving several transmembrane and cytoplasmic proteins, which eventually affects the concentration of a pivotal response regulator, CheY. This protein rapidly diffuses inside the cell and interacts with the proteins of the flagellar motors, thus inducing clockwise (CW) and counterclockwise (CCW) rotation of each flagellum. When flagella are turning CW, they are uncoordinated and the bacterium performs a *tumbling* movement, while if they are all turning CCW, they form a bundle and get coordinated, thus allowing the cell to swim directionally (the so-called *running* movement). In a homogeneous environment, bacteria perform a temporal sampling of their surroundings by moving with a random walk, that is caused by a high switch frequency of the flagellar motors rotations, that alternate rapid tumbling with short running movements. In the presence of a ligand concentration gradient, instead, bacteria carry out directional swimming toward/against the attractants/repellents, by reducing the switch frequency of flagella rotations, that results in longer running movements. If the ligand concentration remains constant in time, then the switch frequency is reset to the prestimulus level, therefore realizing an *adaptation* of the chemotactic response to the change in ligand concentration. In what follows, we consider the chemosensory system of *E. coli* bacteria, in response to attractant chemicals.

The chemotactic pathway, depicted in Fig. 3.1, has been well characterized from a molecular point of view [5, 24, 25]. External signals are detected by transmembrane methyl-accepting proteins (MCPs), which are linked to cytoplasmic histidine protein kinases (CheA) by means of scaffold proteins (CheW). These three proteins constitute the sensor module (i.e., the receptor complexes) of the whole pathway; each protein occurs as a dimer in every receptor complex. The role of CheA is to transduce the presence of an external ligand toward the inside of the cell, by phosphorylating two cytoplasmic proteins, called CheY and CheB. The transfer of the phosphoryl group to these proteins is more probable—that is, the activity of CheA is stronger—in absence of external ligands. CheY and CheB compete for the binding to CheA, but the phosphotransfer to CheY is faster than to CheB [25]; this fact assures that the proper chemotactic response can be generated before the process of adaptation occurs, as explained hereafter. CheY is the response regulator protein which, after being phosphorylated, interacts with the proteins FliM of the flagellar motors, inducing the CW rotation of the flagellum and the tumbling movements (FliM is a key component of the processes that stands *downstream* of the chemotaxis signaling, and therefore will not be explicitly included in our model). In presence of external ligands, the activity of CheA is reduced: the concentrations of phosphorylated CheY diminishes, its interaction with the flagellar motors is reduced, the CCW rotation is switched on, and bacteria can perform longer running movements. The termination of this signal transduction process is mediated by another



**Fig. 3.1** Signal transduction pathway in bacterial chemotaxis: *solid arrows* indicate enzyme-catalyzed reactions, *dashed arrows* indicate autocatalysis;  $\text{CH}_3$  denotes the methyl group, P the phosphoryl group (the dimensions of components are not scaled)

cytoplasmic protein, CheZ, which acts as an allosteric activator of CheY dephosphorylation. Concurrently to the processes involving CheY, the chemosensory system possesses an adaptation response which depends on the methylation level of the receptors. Methylation reactions are modulated by the coordinated interplay between proteins CheR and CheB. Up to 4–6 methyl groups are constantly transferred to the cytoplasmic domain of MCPs by the constitutively active methyltransferases CheR. On the other side, the demethylation of MCPs occurs by means of the phosphorylated form of the methylesterase CheB. The methylation state of MCPs also intervene on the regulation of CheA: when MCPs are highly methylated, CheA is more active; when MCPs are unmethylated, the activity of CheA is reduced. In the latter case, also the concentrations of phosphorylated CheB diminishes, and this in turn lets the methylation state of MCPs increase, with a consequent renewed activity of CheA, and so on through a continuous feedback control. Therefore, the cell is able to adapt to environmental changes and return to the random walk sampling when the concentration gradient of the attractant remains constant in time. This feedback mechanism also allows bacteria to widen the range of ligand concentration to which they can respond, making them very sensible to low environmental variations.

**Table 3.1** Molecular amounts of the species initially present in the multiset  $M_1$ 

Molecular species	Initial amount
2MCP	4000
2CheW	4000
2CheA	4000
CheY	17000
CheZ	12000
CheR	200
CheB	1700
ATP	$1.2 \cdot 10^6$

These values, corresponding to the amounts of the 7 elementary chemotactic proteins initially occurring in the system, have been taken from [27]. The amount of ATP is kept constant during the evolution of the system

### 3.3.2 A Mechanistic Model

In this section, we present the mechanistic model of bacterial chemotaxis that has been formalized by means of a  $\tau$ -DPP composed of a single volume, 32 molecular species and 62 reactions [26].

The  $\tau$ -DPP  $\gamma_{SV}$  describing the chemotaxis pathway is defined as

$$\gamma_{SV} = (V_1, \mu, \mathcal{S}, M_1, R_1, C_1),$$

where:

- $V_1$  is the volume representing the bacterium cytoplasm;
- $\mu = [1]_1$ ;
- $\mathcal{S} = \{ 2\text{MCP}, 2\text{CheW}, 2\text{MCP}::2\text{CheW}, 2\text{CheA}, 2\text{MCP}::2\text{CheW}::2\text{CheA}, \text{ATP}, 2\text{MCP}::2\text{CheW}::2\text{CheAp}, \text{CheY}, \text{CheYp}, \text{CheB}, \text{CheBp}, \text{CheZ}, \text{lig}, \text{lig}::2\text{MCP}::2\text{CheW}::2\text{CheA}, \text{CheR}, \text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA}, \text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheAp}, 2\text{MCP}^m::2\text{CheW}::2\text{CheA}, 2\text{MCP}^m::2\text{CheW}::2\text{CheAp} \}$  for  $m = 0, \dots, 4$ ;
- $M_1$  is the set of the initial amounts of molecular species (given in Table 3.1);
- $R_1$  is the set of reactions, reported in Table 3.2;
- $C_1$  is the set of stochastic constants associated to  $R_1$ , reported in the caption of Table 3.2.

The initial multiset  $M_1$  occurring in the system is reported in Table 3.1, the amounts of all other molecular species present in  $\mathcal{S}$  are initially equal to zero, as they are produced by mimicking the formation and dissociation of protein complexes, and by describing the phosphorylation/dephosphorylation of cytoplasmic proteins and the methylation/demethylation of MCPs, in both the conditions of presence and absence of external ligands.

As described in Sect. 3.2, each reaction in the model is given in the form “reagents  $\rightarrow$  products”, where the notation  $X + Y$  is used to represent a molecular interaction

**Table 3.2** The 62 reactions of the single volume model of bacterial chemotaxis

	Reagents	Products	Methylation state
1	$2\text{MCP}^m + 2\text{CheW}$	$2\text{MCP}^m::2\text{CheW}$	$m = 0$
2	$2\text{MCP}^m::2\text{CheW}$	$2\text{MCP}^m + 2\text{CheW}$	$m = 0$
3	$2\text{MCP}^m::2\text{CheW} + 2\text{CheA}$	$2\text{MCP}^m::2\text{CheW}::2\text{CheA}$	$m = 0$
4	$2\text{MCP}^m::2\text{CheW}::2\text{CheA}$	$2\text{MCP}^m::2\text{CheW} + 2\text{CheA}$	$m = 0$
5–8	$2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheR}$	$2\text{MCP}^{m+1}::2\text{CheW}::2\text{CheA} + \text{CheR}$	$m = 0, \dots, 3$
9–12	$2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheBp}$	$2\text{MCP}^{m-1}::2\text{CheW}::2\text{CheA} + \text{CheBp}$	$m = 1, \dots, 4$
13–17	$2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{ATP}$	$2\text{MCP}^m::2\text{CheW}::2\text{CheAp}$	$m = 0, \dots, 4$
18–22	$2\text{MCP}^m::2\text{CheW}::2\text{CheAp} + \text{CheY}$	$2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheYp}$	$m = 0, \dots, 4$
23–27	$2\text{MCP}^m::2\text{CheW}::2\text{CheAp} + \text{CheB}$	$2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheBp}$	$m = 0, \dots, 4$
28–32	$\text{lig} + 2\text{MCP}^m::2\text{CheW}::2\text{CheA}$	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA}$	$m = 0, \dots, 4$
33–37	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA}$	$\text{lig} + 2\text{MCP}^m::2\text{CheW}::2\text{CheA}$	$m = 0, \dots, 4$
38–41	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheR}$	$\text{lig}::2\text{MCP}^{m+1}::2\text{CheW}::2\text{CheA} + \text{CheR}$	$m = 0, \dots, 3$
42–45	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheBp}$	$\text{lig}::2\text{MCP}^{m-1}::2\text{CheW}::2\text{CheA} + \text{CheBp}$	$m = 1, \dots, 4$
46–50	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{ATP}$	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheAp}$	$m = 0, \dots, 4$
51–55	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheAp} + \text{CheY}$	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheYp}$	$m = 0, \dots, 4$
56–60	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheAp} + \text{CheB}$	$\text{lig}::2\text{MCP}^m::2\text{CheW}::2\text{CheA} + \text{CheBp}$	$m = 0, \dots, 4$
61	$\text{CheYp} + \text{CheZ}$	$\text{CheY} + \text{CheZ}$	$m = 0, \dots, 4$
62	$\text{CheBp}$	$\text{CheB}$	

The values contained in  $C_1$ , corresponding to the stochastic constants of the reactions in  $R_1$ , are:  $c_1 = 0.1$ ,  $c_2 = 0.01$ ,  $c_3 = 0.1$ ,  $c_4 = 0.02$ ,  $c_5 = 0.325$ ,  $c_6 = 0.29$ ,  $c_7 = 0.165$ ,  $c_8 = 0.05$ ,  $c_9 = 0.0044$ ,  $c_{10} = 0.0175$ ,  $c_{11} = 0.0306$ ,  $c_{12} = 0.035$ ,  $c_{13} = 5.0 \cdot 10^{-7}$ ,  $c_{14} = 7.0 \cdot 10^{-6}$ ,  $c_{15} = 2.8 \cdot 10^{-5}$ ,  $c_{16} = 5.0 \cdot 10^{-5}$ ,  $c_{17} = 6.8 \cdot 10^{-5}$ ,  $c_{18} = 5.0 \cdot 10^{-4}$ ,  $c_{19} = 0.0035$ ,  $c_{20} = 0.014$ ,  $c_{21} = 0.025$ ,  $c_{22} = 0.0336$ ,  $c_{23} = 2.0 \cdot 10^{-4}$ ,  $c_{24} = 0.0014$ ,  $c_{25} = 0.0056$ ,  $c_{26} = 0.01$ ,  $c_{27} = 0.0135$ ,  $c_{28} = 0.6$ ,  $c_{29} = 0.8$ ,  $c_{30} = 1.0$ ,  $c_{31} = 1.2$ ,  $c_{32} = 1.4$ ,  $c_{33} = 15.0$ ,  $c_{34} = 15.0$ ,  $c_{35} = 15.0$ ,  $c_{36} = 15.0$ ,  $c_{37} = 15.0$ ,  $c_{38} = 4.0 \cdot 10^{-4}$ ,  $c_{39} = 3.75 \cdot 10^{-4}$ ,  $c_{40} = 3.5 \cdot 10^{-4}$ ,  $c_{41} = 2.125 \cdot 10^{-4}$ ,  $c_{42} = 6.0 \cdot 10^{-4}$ ,  $c_{43} = 0.0044$ ,  $c_{44} = 0.0175$ ,  $c_{45} = 0.0343$ ,  $c_{46} = 1.0 \cdot 10^{-8}$ ,  $c_{47} = 5.0 \cdot 10^{-7}$ ,  $c_{48} = 7.0 \cdot 10^{-6}$ ,  $c_{49} = 2.8 \cdot 10^{-5}$ ,  $c_{50} = 5.0 \cdot 10^{-5}$ ,  $c_{51} = 1.0 \cdot 10^{-5}$ ,  $c_{52} = 5.0 \cdot 10^{-4}$ ,  $c_{53} = 0.0035$ ,  $c_{54} = 0.014$ ,  $c_{55} = 0.03$ ,  $c_{56} = 1.0 \cdot 10^{-5}$ ,  $c_{57} = 2.0 \cdot 10^{-4}$ ,  $c_{58} = 0.0014$ ,  $c_{59} = 0.0056$ ,  $c_{60} = 0.0112$ ,  $c_{61} = 0.0080$ ,  $c_{62} = 1.0$ . All values are in  $\text{s}^{-1}$  time units

between species  $X$  and  $Y$ , while  $X::Y$  denotes that  $X$  and  $Y$  are chemically bound in the formation of a complex (see Table 3.2). The phosphorylated form of species  $X$ , with  $X \in \{\text{CheA}, \text{CheB}, \text{CheY}\}$ , is denoted by  $X_p$ , while the methylation state of receptor MCP is denoted by  $\text{MCP}^m$ , for  $m = 0, \dots, 4$  (that is, five methylation states are considered).

The reactions in  $R_1$  have been “clustered” in Table 3.2 according to the different molecular interaction mechanisms that they describe:

- association of the three dimers ( $2\text{MCP}$ ,  $2\text{CheW}$  and  $2\text{CheA}$ ) constituting each ternary receptor complex (reactions 1–4);
- binding and unbinding of ligand molecules to the receptor complex in the five methylation states (reactions 28–32 and 33–37, respectively);
- methylation and demethylation of MCPs, in absence and in presence of ligand molecules (reactions 5–8, 9–12 and 38–41, 42–45, respectively);
- autophosphorylation of CheA in the five methylation states of MCPs, in absence and in presence of ligand molecules (reactions 13–17 and 46–50, respectively);
- phosphotransfer to CheY in the five methylation states of MCPs, in absence and in presence of ligand molecules (reactions 18–22 and 51–55, respectively);
- phosphotransfer to CheB in the five methylation states of MCPs, in absence and in presence of ligand molecules (reactions 23–27 and 56–60, respectively);
- dephosphorylation of CheY<sub>p</sub> and CheB<sub>p</sub> (reactions 61–62).

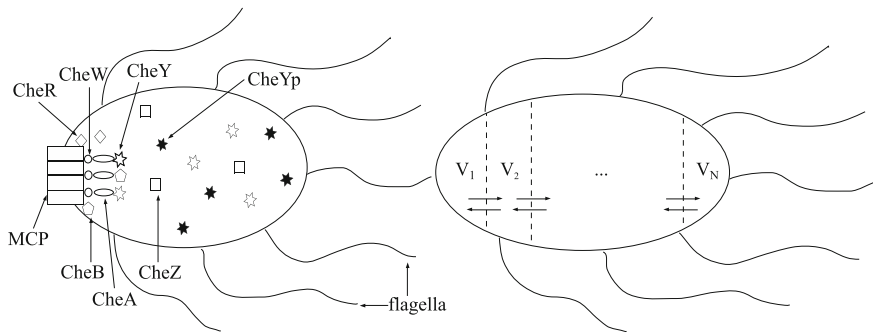
According to literature, the ternary receptor complex  $2\text{MCP}^m::2\text{CheW}::2\text{CheA}$  is assumed to be stable for the duration of the signal transduction process [28]; moreover, the synthesis and degradation rates of all chemotactic proteins are assumed to occur at a much slower scale than the chemotactic response (hence, the reactions corresponding to these processes have not been included in the model).

A stochastic constant is associated to each reaction: it is needed to evaluate the probability of that reaction to occur when performing stochastic simulations, as explained in [14]. The stochastic constants used for all simulations presented in Sect. 3.4 are those reported in the caption of Table 3.2 (all values are expressed in  $\text{s}^{-1}$ ). Some of these values have been derived from literature [29], and some tuned to account for the following biological features [30, 31]:

- (1) the binding affinity of the ligand is directly proportional to the methylation state of MCPs;
- (2) the ligand-receptor binding reactions occur at a faster rate with respect to phosphorylation and methylation/demethylation reactions;
- (3) the methylation and demethylation activities of CheR and CheB<sub>p</sub> are, respectively, inversely and directly proportional to the methylation state of MCPs;
- (4) the rate of phosphotransfer from CheA to CheY and to CheB depends on the rate of autophosphorylation of CheA.

According to these constraints, which set the relative magnitude of some constants with respect to others, the estimation of the unavailable constants has been performed by testing the effect of a range of values for each constant within every module of the model. By a module we mean a group of reactions corresponding to a specific process,





**Fig. 3.2** The signal transduction pathway in bacterial chemotaxis. *Left* chemotactic proteins and diffusion of CheYp from receptors site to flagella. *Right* formal representation of the system with the partition into  $N$  virtual volumes

such as, e.g., reactions 1–4 that describe the formation of the receptor complexes. Within this range, the value for each constant chosen in the end is the one that yields a good reproduction of the expected behavior of the biological subsystem described by that module. Then, every other module has been sequentially added to the previous ones, following the same iterative process, to perform a comprehensive and biologically correct simulation of the whole pathway.

### 3.3.3 Multivolume Model: Diffusion in a Signal Transduction Pathway

To the aim of modeling the diffusive processes in bacterial chemotaxis, here we present an extension of the single volume model described in Sect. 3.3.2 [32]. In that model we did not take into account the diffusion of CheYp [33], which moves from the place where it is phosphorylated (close to one polar region of the cell) to the location of flagella (randomly and asymmetrically placed along the bacterial surface, but more frequently occurring at the distal region—see Fig. 3.2, left side). Similarly, also the non phosphorylated form of this protein, CheY, can diffuse from the anterior to the distal end of the cell, and back. The dephosphorylation of CheYp is aided by protein CheZ, which can also diffuse throughout the bacterial cytoplasm [34, 35].

To the purpose of mimicking diffusion events in cellular systems, we consider the following procedure within  $\tau$ -DPP for the evaluation of the stochastic constants associated to communication (diffusion) reactions. Let us denote by  $\Omega \subseteq \mathbb{R}^3$  the entire bacterial volume (cytoplasm), and consider a subdivision of  $\Omega$  into  $N$  adjacent and non overlapping subvolumes  $V_k$ ,  $k = 1, \dots, N$ , each one described by a characteristic length  $h_k$  (that is,  $V_k = h_k^3$ ), such that  $\Omega = \bigoplus_k V_k$ . Each  $V_k$  represents a “virtual volume”, that is, a space that does not correspond to a real sub-compartment of the bacterial cytoplasm  $\Omega$ , but whose definition is necessary in order to describe

the local distributions of chemicals and to handle the diffusion of these chemicals between any couple of neighboring subvolumes. The partition of  $\Omega$  into virtual volumes represents a natural extension of the master equation approach [16] to heterogeneous (space) systems, whereby each virtual volume can be assumed to be homogeneous.

In these conditions, for each species  $X_i$  that occurs in the virtual volume  $V_k$  and that is free to diffuse, it is possible to define a mean jump frequency, given by  $\tilde{D}_{i,k} = (6D_i)/h_k^2$  [15], which allows to connect the microscopic description of the master equation with the macroscopic Fick's diffusion coefficient  $D_i$  of species  $X_i$ , and to verify if the well-stirred condition still holds. The latter requirement is equivalent to impose that, *inside* each virtual volume, the diffusion time  $\tau_{\text{diff}} \approx h_k^2/(6D_i)$  of species  $X_i$  is much smaller than the reaction waiting times  $\tau$  [36]; this condition should also be granted for the rate of diffusion of each species *between* neighboring subvolumes. This observation allows to derive the expression to evaluate the value of the stochastic constant associated to the communication reactions of species  $X_i$ , that is,  $c_i^{\text{diff}} \propto D_i/h_k^2$ . Therefore, for each diffusing species  $X_i \in \{\text{CheZ}, \text{CheY}, \text{CheYp}\}$  in the bacterial chemotaxis pathway, we consider the length of bacterial cell  $L = 3.3 \mu\text{m}$  and the values of diffusion coefficients  $D_{\text{CheZ}} = 0.66 \mu\text{m}^2\text{s}^{-1}$ ,  $D_{\text{CheY}} = D_{\text{CheYp}} = 1.26 \mu\text{m}^2\text{s}^{-1}$  reported in [33]. We obtain  $h_k = L/N$  and, assuming for instance  $N = 6$ , we derive that  $c_{\text{CheZ}}^{\text{diff}} \propto 2.18 \text{ s}^{-1}$  and  $c_{\text{CheY}}^{\text{diff}} = c_{\text{CheYp}}^{\text{diff}} \propto 4.17 \text{ s}^{-1}$ .

To account for diffusion processes, the single volume model described in Sect. 3.3.2 has been modified as follows. The internal reactions corresponding to protein-protein interactions and to the sensing of the external ligand (Table 3.2) are all assumed to occur inside the virtual volume  $V_1$ , which corresponds to the anterior end of the bacterial cell where the chemotactic receptors (MCPs) are placed (see Fig. 3.2, right side). In addition, communication reactions are included in each couple of neighboring virtual volumes  $V_k, V_{k+1}$ , for every  $k = 1, \dots, N-1$ , in order to allow the diffusion of the three species CheZ, CheY, CheYp throughout the entire volume. We also assume that CheZ is uniformly distributed within the entire bacterial volume, while CheY is initially present only in the virtual volume  $V_1$ , where it is phosphorylated by CheA.

In order to consider different crowding conditions of the bacterial cytoplasm, we have fixed three different sets of stochastic constants (see Table 3.3) for the diffusion reactions of CheZ, CheY, CheYp:

- (1) in *case A*, the values of diffusion constants for CheY, CheYp and CheZ are roughly twice as the derived values  $c_{\text{CheY}}^{\text{diff}}, c_{\text{CheYp}}^{\text{diff}}, c_{\text{CheZ}}^{\text{diff}}$ , as assumed in [33]. This case gives rise to a system dynamics where the mean values of the steady states of CheYp, inside each virtual volume, are very similar each other, up to stochastic fluctuations;
- (2) in *case B*, the values of diffusion constants for CheY, CheYp and CheZ are 1/10 the values of case A, thus mimicking a cytoplasmic condition where diffusion is more limited than case A. This case gives rise to a system dynamics where

the mean values of the steady states of CheYp, inside each virtual volume, are slightly different each other;

- (3) in *case C*, the values of diffusion constants for CheY, CheYp and CheZ are 1/40 the values of *case A*, thus mimicking a crowded cytoplasm where diffusion is strongly inhibited by the presence of obstacles. This case gives rise to a system dynamics where the mean values of the steady states of CheYp, inside each virtual volume, are markedly different each other (being the difference more evident between the two volumes at the opposite poles of the cell).

The  $\tau$ -DPP  $\Upsilon_{MV}$  describing the chemotaxis pathway along with diffusion processes of the species CheZ, CheY, CheYp is defined as

$$\Upsilon_{MV} = (V_1, \dots, V_N, \mu, \mathcal{S}, M_1, \dots, M_N, R_1, \dots, R_N, C_1, \dots, C_N),$$

where:

- $V_1, \dots, V_N$  are the virtual volumes representing the bacterium cytoplasm;
- $\mu = [1]_1 \dots [N]_N$  is the membrane structure where virtual volumes are arranged as a chain graph with undirected edges, that is, only adjacent volumes can communicate. The volume  $V_e$ , namely, the environment which contains volumes  $V_1, \dots, V_N$ , is not formally represented here;
- $\mathcal{S} = \{ 2MCP, 2CheW, 2MCP::2CheW, 2CheA, 2MCP::2CheW::2CheA, ATP, 2MCP::2CheW::2CheAp, CheY, CheYp, CheB, CheBp, CheZ, lig, lig::2MCP::2CheW::2CheA, CheR, lig::2MCP^m::2CheW::2CheA, lig::2MCP^m::2CheW::2CheAp, 2MCP^m::2CheW::2CheA, 2MCP^m::2CheW::2CheAp \}$  for  $m = 0, \dots, 4$ ;
- $M_1, \dots, M_N$  are the sets of initial amounts of molecular species (reported in the following);
- $R_1, \dots, R_N$  are the sets of reactions reported in Tables 3.2 and 3.4, as explained later on;
- $C_1, \dots, C_N$  are the sets of stochastic constants associated to  $R_1, \dots, R_N$ , given in the caption of Tables 3.2 and 3.3.

The initial amounts of molecular species occurring in the multiset  $M_1$  corresponding to volume  $V_1$  are those reported in Table 3.1, except for the value of protein CheZ,

**Table 3.3** Values of the stochastic constants of diffusion reactions, in relation to the three cases considered (A, B, C) and to the number of virtual volumes  $N$  (all values are expressed in  $s^{-1}$ )

		$N = 6$	$N = 12$	$N = 30$	$N = 60$
CheY/CheYp	case A	10	33	208	833
	case B	1	3.3	20.8	83.3
	case C	0.25	0.825	5.2	20.825
CheZ	case A	5	17	110	436
	case B	0.5	1.7	11	43.6
	case C	0.125	0.425	2.75	10.9

which is considered uniformly distributed in all virtual volumes; therefore, its amount inside  $M_1$  is reduced to  $12000/N$ . As in the case of the single volume system  $\Upsilon_{SV}$ , the amount of ATP is kept constant during the evolution of  $\Upsilon_{MV}$  and the initial amounts of all other molecular species are equal to zero. The initial multisets  $M_2, \dots, M_N$  contain only  $12000/N$  monomers of CheZ.

The reactions of the  $\Upsilon_{MV}$  contained in  $R_1$  are those listed in Table 3.2 with the addition of the diffusion reactions for the communications of the species CheZ, CheY, CheYp (Table 3.4). With respect to  $\Upsilon_{SV}$ , in  $\Upsilon_{MV}$  the value of stochastic constant corresponding to reaction 61 in Table 3.2 is reduced to  $8 \times 10^{-6} \text{ s}^{-1}$ . The sets  $R_2, \dots, R_N$  contain the reaction representing the dephosphorylation of CheYp aided by protein CheZ, and the diffusion reactions (Table 3.4), since we suppose that the chemotactic pathway operates only within the volume containing the receptors ( $V_1$  in our model). Note that in the simulation results presented in Sect. 3.4, different values for the diffusion reactions are used (Table 3.3), as explained above.

## 3.4 Results

In this section we present the results obtained from the simulation of the dynamics of  $\Upsilon_{SV}$  and  $\Upsilon_{MV}$  by investigating, in particular, the phosphorylated form of the pivotal protein CheY, and by analyzing the running and tumbling average time in different conditions for both single and multivolume models.

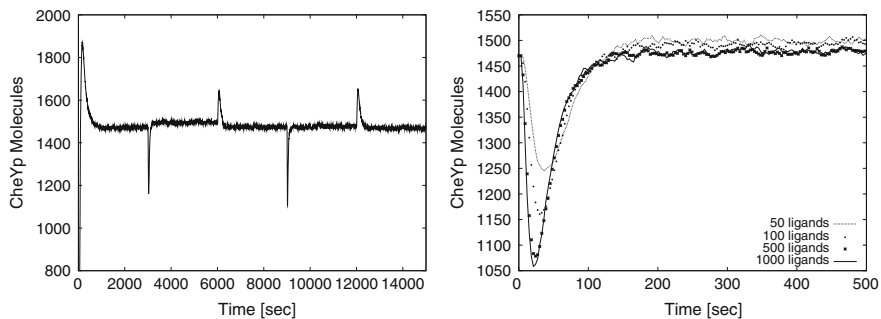
### 3.4.1 Simulations of $\Upsilon_{SV}$

The dynamics of CheYp has been analyzed by considering various conditions, such as the addition and removal of different ligand amounts, distinct methylation states of MCPs and deletion of other chemotactic proteins.

**Table 3.4** Dephosphorylation of CheYp aided by protein CheZ (reaction 1) and diffusion reactions occurring inside the virtual volumes  $V_k$  (where  $1 < k < N$ )

	Reagents	Products	Target volume
1	CheYp + CheZ	CheY + CheZ	$k$
2	CheZ	CheZ	$k + 1$
3	CheZ	CheZ	$k - 1$
4	CheY	CheY	$k + 1$
5	CheY	CheY	$k - 1$
6	CheYp	CheYp	$k + 1$
7	CheYp	CheYp	$k - 1$

It is clear that inside volume  $V_1$  ( $V_N$ ), besides reaction 1, only reactions 2, 4, 6 (3, 5, 7) are present. The stochastic constants associated to diffusion reactions are listed in Table 3.3



**Fig. 3.3** Dynamics of CheYp. *Left* adaptation response to two consecutive stimuli. *Right* comparison of transient and steady state response to different ligand amounts

We start by showing in Fig. 3.3, left side, the response of the system to the addition of two consecutive amounts of external ligand: the first stimulus corresponds to a ligand amount of 100 molecules, added to the system at time  $t = 3,000$  s and removed at time  $t = 6,000$  s, while the second stimulus corresponds to a ligand amount of 500 molecules, added at time  $t = 9,000$  s and removed at time  $t = 12,000$  s. Note that, since the amount of CheYp is equal to 0 at the beginning of the simulation, its dynamics initially shows a marked increase which then reaches a steady state level, due to the counteraction of CheZ, CheR and CheB. Starting from this level, the addition of ligands has been simulated by changing the amount of the species  $lig$  contained within the multiset  $M_1$  of  $\mathcal{T}_{SV}$  from 0 to 100 molecules for the first stimulus (from 0 to 500 molecules for the second stimulus), thus mimicking the environmental situation where the bacterium encounters a different concentration of attractant molecules. Vice versa, the removal of ligands has been simulated by putting the value of  $lig$  back to 0 in  $M_1$ . In the time interval between the addition and the removal of each ligand stimulus, the amount of ligand molecules has been kept at the constant value of 100 molecules for the first stimulus and 500 molecules for the second, thus mimicking the presence of an environmental homogeneous concentration. These two perturbations have been done in order to test the adaptation capabilities of the system. In both cases, we can see that the system is able to respond to a step-increase of the ligands by achieving a sharp and fast decrease in CheYp (that is, the negative peaks at time instants  $t = 3,000$  s and  $t = 9,000$  s). Immediately after this transient, the amount of CheYp returns to a steady state value, which differs from the prestimulus level only for a few tens of molecules, at most, according to the amount of added ligand. In this phase, the bacterium is returning to the prestimulus switching and thus to the random walk sampling of its surroundings. When the ligand is removed, CheYp shows another transient behavior, corresponding to a sharp and fast increase of its amount. After this second transient, the amount of CheYp correctly returns to the prestimulus steady state level. These results are in line with experimental observations, as well as with previous simulations of CheYp dynamics carried out using different models of bacterial chemotaxis (see, e.g., [10, 11, 37]).

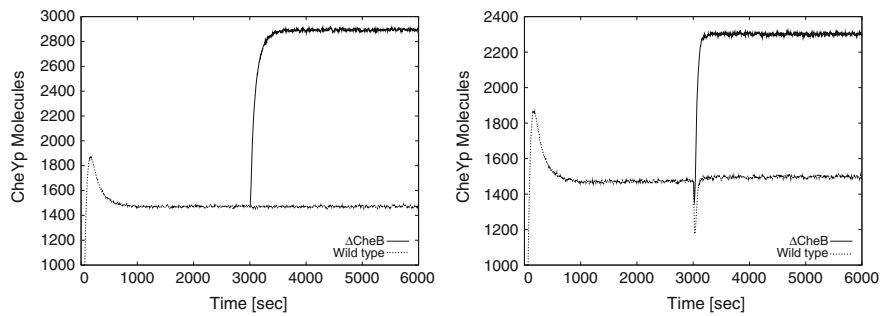
**Table 3.5** Steady state values and minimum/maximum transient values of CheYp after addition and removal of distinct ligand amounts

Ligand amount	SS1	Min	SS2	Max	SS3
50 molecules	1486.7	1245.4	1500.9	1626.0	1474.7
100 molecules	1486.7	1160.7	1495.1	1645.4	1474.3
500 molecules	1486.7	1078.4	1481.4	1653.2	1469.4
1000 molecules	1486.7	1058.6	1478.2	1665.8	1474.7

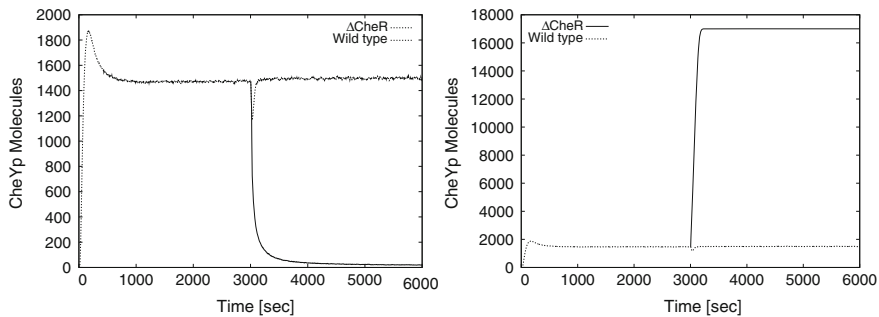
In Fig. 3.3, right side, we compare the transients and the steady state levels reached by CheYp after the addition of distinct ligand amounts to the multiset  $M_1$ . This figure shows that the response magnitude at steady state and the adaptation time of CheYp is only slightly sensitive to the ligand amount, being the relative differences less than a few tens of molecules and less than a few seconds, respectively. The mean values of the steady state of CheYp before the stimulus (SS1), after the ligand addition (SS2) and after the ligand removal (SS3) are reported in Table 3.5, together with the values of its minimum and maximum values immediately after the ligand addition and removal (Min and Max, respectively), for the four ligand amounts (50, 100, 500, 1,000 molecules) considered in the right side of Fig. 3.3.

In Fig. 3.4 we show how the dynamics of CheYp changes when CheB is deleted from the multiset  $M_1$  at time  $t = 3,000$  s, in both conditions of absence of external ligands (left side) and of presence of 100 molecules of ligand (right side) added to  $M_1$  at time  $t = 3,000$  s. CheB is the methylesterase that, once being phosphorylated by CheA, increases the methylation state of MCPs, thus keeping CheA more active. This, in turn, causes an increase in the amount of CheYp, which is evident from its new steady state level reached after CheB deletion, and also from its less negative transient decrease after ligand addition.

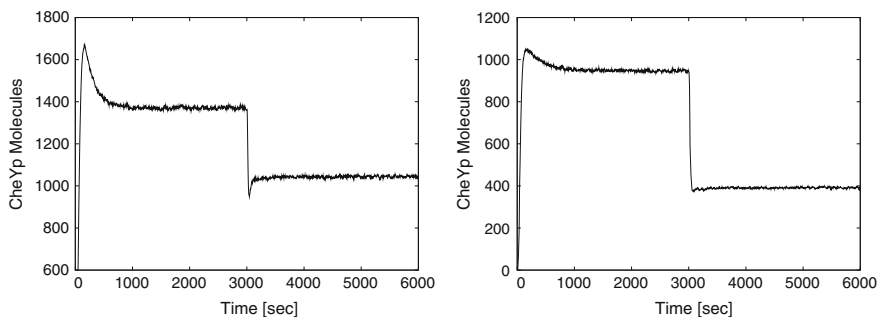
Similarly, in Fig. 3.5 we show the dynamics of CheYp when either CheR (left side) or CheZ (right side) are deleted from the multiset  $M_1$  at time  $t = 3,000$  s,



**Fig. 3.4** Comparison of dynamics of CheYp in normal condition and after deletion of CheB at  $t = 3,000$  s, without ligand (left) and with simultaneous addition of 100 ligand molecules (right)



**Fig. 3.5** Comparison of dynamics of CheYp in normal condition and after deletion of CheR (*left*) and CheZ (*right*) at  $t = 3,000$  s, both simulated with a simultaneous addition of 100 ligand molecules



**Fig. 3.6** Dynamics of CheYp when only 3 (*left*) and 2 (*right*) methylation states are active

simultaneously to the addition of 100 ligand molecules. When CheR is deleted, its methyltransferase activity is silenced, the MCPs are no more methylated, and hence the amount of CheYp tends to zero. On the contrary, when CheZ is deleted, all CheY molecules always remain phosphorylated. For the sake of completeness, we have also simulated the dynamics of CheYp when either CheB, CheR or CheZ are deleted from the multiset  $M_1$  at time instant  $t = 0$ , in order to have a comparison about the initial temporal evolution of CheYp and the steady state levels it can reach. In these conditions, the model correctly simulates [6, 38, 39] a very low production of CheYp when CheR is deleted, and an increased production (albeit with different magnitudes) when either CheB or CheZ are deleted (data not shown).

Finally, in Fig. 3.6 we compare the dynamics of CheYp in response to the addition of 100 ligand molecules to  $M_1$  at  $t = 3,000$  s, when only 3 (left side) or 2 (right side) methylation states of the receptors are allowed. In practice, this is achieved by initially putting to zero the values of the stochastic constants of methylation and demethylation reactions for levels  $m = 4$  and  $m = 3$ , respectively. In both cases, we see that the system is not able to adapt, as the steady state level of CheYp reached after the addition of the ligand to the multiset  $M_1$  is substantially lower than the steady state when all methylation levels are activated.

The outcomes of the stochastic simulations reported in this section provide a preliminary validation of the model described by means of the  $\tau$ -DPP  $\Upsilon_{SV}$  presented in Sect. 3.3.2, as the dynamics of CheYp under different conditions of the chemotactic pathway are qualitatively analogous to experimental evidences. Additional *ad hoc* experiments would be necessary to fully verify the simulation results and the predictions achieved by our model, as also discussed further on.

### 3.4.2 *The Interplay Between Stochastic Fluctuations and the Number of Bacterial Flagella in $\Upsilon_{SV}$*

In this section we make use of the simulations based on our single volume model of chemotaxis to investigate the interplay between stochastic fluctuations of CheYp and the number of flagella occurring on the cell, in order to outline the influence of synchronization of flagellar motors on the swimming behavior and on the adaptation mechanism of the bacterium to the environmental changes. To this aim, we consider the dynamics of CheYp at steady state, as well as its transient step-decrease that takes place immediately after the chemotactic stimulus. In both cases, we are interested in devising the time periods during which the cell performs either a running or a tumbling motion. In particular we assume that: (1) the time spent in alternating CW and CCW rotations during the steady state corresponds to the random walk sampling of the environment—where we expect more time spent in tumbling than in running motions; (2) the time required to return to the prestimulus level of CheYp (that is, the transient response immediately after the ligand addition) corresponds to the chemotactic adaptation time—where we expect a much longer and uninterrupted time interval of running motion with respect to the steady state condition.

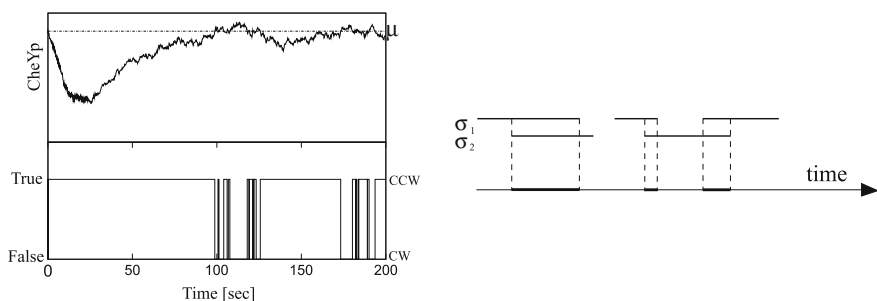
As explained in Sect. 3.3.1, a running motion requires that all flagella are simultaneously synchronized in a CCW rotation—which occurs when CheYp is not interacting with the proteins FlhM of the flagellar motors, that is, when its intracellular concentration diminishes with respect to a reference value. To distinguish between the CW and CCW rotations of a *single* flagellum, we assume that the flagellar motor switch is sensitive to a threshold level of CheYp, that is hereby evaluated as the mean value of CheYp at steady state (see also [8], where a similar approach of threshold-crossing mechanism for motor switching was tested, albeit that work considered only a unique flagellum and did not propose any investigation on the simultaneous coordination of many flagella). When the amount of CheYp is below this threshold, each flagellum is rotating CCW, while when the amount of CheYp is above the threshold, each flagellum is rotating CW. In what follows, we make a one-to-one correspondence between the behavior of a single flagellum and a temporal evolution of CheYp generated by one run of  $\tau$ -DPP, that is, we consider a different and independent stochastic simulation for each and every flagellum (albeit starting from the same initial conditions for the whole system). In other words, we assume that flagella are independent of each other—as no molecular interactions



between them have been evidenced in bacterial cells. Nonetheless, they all overlook on the same intracellular ambient, that is, they are all subject to the same temporal evolution of CheYp, apart from the stochastic noise existing among independent simulations. In order to determine the synchronization of all flagella that will induce a running motion of the bacterium, we therefore need to identify the time instants when *all* flagella are rotating CCW, that is, to select the time intervals when *all* the temporal evolutions of CheYp are below the fixed threshold. We remark here that a similar analysis would be impossible using the outcome of deterministic simulations, therefore supporting our choice in the definition of stochastic models.

Formally, we proceed as follows. Let  $n = 1, \dots, 10$  be the number of flagella  $f_1, \dots, f_n$  whose influence we want to test, and let  $\sigma_i, i = 1, \dots, n$ , be the time series of CheYp amount (generated by a single simulation of the dynamics of  $\Upsilon_{SV}$ ) associated to each  $f_i$ . For any fixed value of  $n$ , the total time of the simulation considered to generate the dynamics of CheYp is the same for all  $\sigma_i$ . This simulation time, hereby denoted by  $\Delta t_{sim}$ , is chosen long enough to have a meaningful evaluation of the mean intervals of running and tumbling in the analysis performed below (e.g.,  $\Delta t_{sim} = 40,000, 60,000, 120,000$  s for  $n = 1, 5, 10$ , respectively). The threshold for CheYp is evaluated in the following way: we choose an initial time instant at the steady state level—distant enough from the step decrease of CheYp after ligand addition, i.e., 1,000 s afterward—and then, starting from this instant and till the end of  $\Delta t_{sim}$ , we calculate the mean value  $\mu_i = \langle \sigma_i \rangle$  for each  $\sigma_i$ . Then, we define a common threshold  $\mu$  for all flagella, that is,  $\mu = \frac{1}{n} \sum_{i=1, \dots, n} \mu_i$ . This threshold is considered as the reference value, also for the portion of the CheYp dynamics corresponding to the transient decrease after ligand addition. In Fig. 3.7, top panel on the left side, we show a part of  $\Delta t_{sim}$  over a single simulation of CheYp, where both the initial transient response and the stochastic fluctuations around the threshold are evidenced. For all the results discussed below, the different values of  $\mu$  have been found to be approximately equal to 1,480 molecules.

The next step consists in detecting, for each  $f_i$ , the time intervals during which the amount of CheYp remains below  $\mu$ , each one of these intervals corresponding to a CCW rotation time interval of that flagellum. Namely, for each  $\sigma_i$  we identify the time



**Fig. 3.7** Threshold-crossing intervals in stochastic fluctuations of CheYp (*left*) and synchronization of running motion between two flagella (*right*)

intervals  $\Delta t_{true} \subseteq \Delta t_{sim}$  such that  $\Delta t_{true} = \{t \in \Delta t_{sim} \mid \sigma_i(t) - \mu \leq 0\}$ . Note that this simple mechanism of single threshold-crossing could be extended to consider more complex situations—e.g., a double threshold-crossing mode can be assumed—whereby one simply asks for analogous conditions to be satisfied. Similarly, for each  $\sigma_i$  we can determine the complementary time intervals  $\Delta t_{false} \subseteq \Delta t_{sim}$  such that  $\Delta t_{false} = \{t \in \Delta t_{sim} \mid \sigma_i(t) - \mu > 0\}$ ; these intervals correspond to the time that each flagellum  $f_i$  spends in a CW rotation. Stated in other terms, we can associate to each  $\sigma_i$  a function  $CCW_{\sigma_i} : \Delta t_{sim} \rightarrow \{true, false\}$  defined as:

$$CCW_{\sigma_i}(t) = \begin{cases} true & \text{if } \sigma_i(t) - \mu \leq 0, \\ false & \text{otherwise.} \end{cases}$$

In Fig. 3.7, bottom panel on the left side, we show the values of this function for the CheYp simulation given in the upper panel. As it can be seen at a glance, the transient response after ligand addition (when the amount of CheYp is initially below  $\mu$ ) corresponds to a longer and uninterrupted interval of CCW rotation of that flagellum.

Once that the set of all  $\Delta t_{true}$  intervals—or, equivalently, of all functions  $CCW_{\sigma_i}$ —have been determined for each flagellum, the synchronization for any given number  $n$  of flagella can be evaluated. To this aim, let us define  $\mathcal{T}_{sync}^n = \{t \in \Delta t_{sim} \mid CCW_{\sigma_i}(t) = true \text{ for all } i = 1, \dots, n\}$ .  $\mathcal{T}_{sync}^n$  is the set of all times during which *all* time series  $\sigma_i$  are below the threshold  $\mu$ , that is, the time intervals during which *all* flagella are rotating CCW. More precisely, we identify these intervals as the running motion of the bacterium, i.e.,  $\mathcal{T}_{sync}^n$  corresponds to the time of directional swimming—when all flagella are coordinated in a bundle. As an example, in Fig. 3.7, right side, we represent the functions  $CCW_{\sigma_i}(t) = true$  for  $i = 1, 2$ , and the corresponding set  $\mathcal{T}_{sync}^n$ ,  $n = 2$ . The complementary set,  $\mathcal{T}_{unsync}^n = \Delta t_{sim} \setminus \mathcal{T}_{sync}^n$ , corresponds instead to tumbling motion—when at least one flagellum (over the set of  $n$  flagella considered time by time) is rotating CW. Namely,  $\mathcal{T}_{unsync}^n = \{t \in \Delta t_{sim} \mid \text{there exists } i = 1, \dots, n \text{ such that } CCW_{\sigma_i}(t) = false\}$ .

We are now interested in understanding if and how the time intervals within the set  $\mathcal{T}_{sync}^n$  are influenced by the increase of  $n$ . We have performed this analysis over a set of 10 distinct *in silico* experiments (each one corresponding to a cell with  $n$  flagella, with  $n = 1, \dots, 10$ ), and then we have evaluated the mean values of the following three parameters:

1. the time intervals corresponding to a running motion of the bacterium,  $\langle \Delta t_{run} \rangle$ , when all flagella are rotating CCW (that is, the time intervals when all time series  $\sigma_i$  are below  $\mu$ );
2. the time intervals corresponding to a tumbling motion of the bacterium,  $\langle \Delta t_{tumb} \rangle$ , when at least one flagellum over the  $n$  flagella is rotating CW (that is, the time intervals when at least one time series  $\sigma_i$  is above  $\mu$ );
3. the time intervals corresponding to the transient decrease of CheYp after ligand addition,  $\langle \Delta t_{adapt} \rangle$ , that is, the adaptation time during which the bacterium is performing a longer running motion.

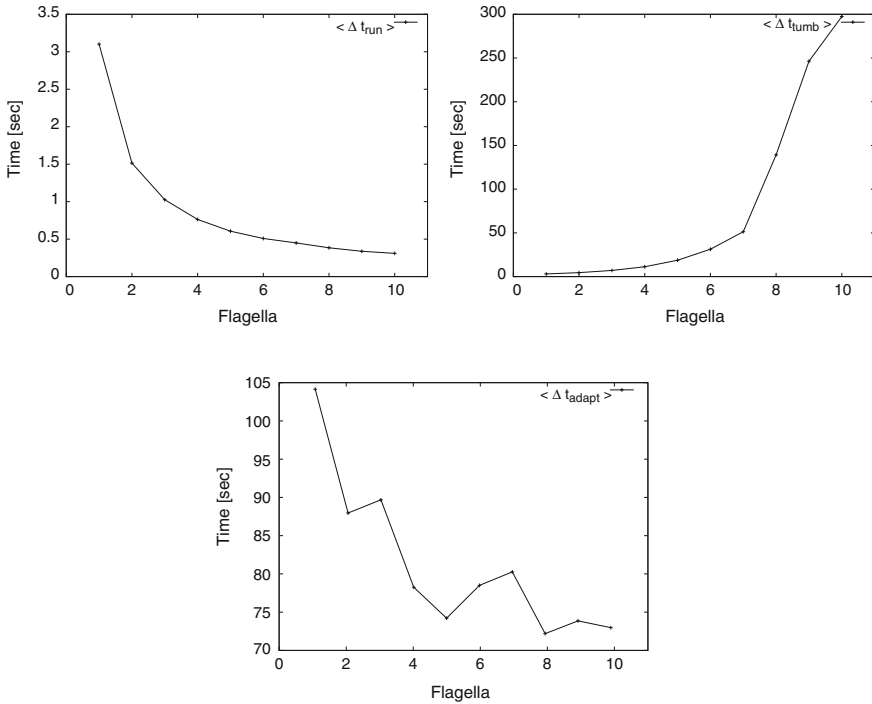
**Table 3.6** Values of mean time intervals for running, tumbling and adaptation

$n$	$\langle \Delta t_{run} \rangle$ (s)	$\langle \Delta t_{tumb} \rangle$ (s)	$\langle \Delta t_{run} \rangle / \langle \Delta t_{tumb} \rangle$	$\langle \Delta t_{adapt} \rangle$ (s)
1	3.102	3.062	1.013	104.0
5	0.606	18.73	0.032	73.48
10	0.310	297.4	0.001	72.22

The results for  $\langle \Delta t_{run} \rangle$  are reported in Fig. 3.8, top left, where we can see that the mean time intervals of running motion are very short, and their values decrease in a (qualitative) exponential way as the number  $n$  of flagella increases, as expected. Similarly, the results for  $\langle \Delta t_{tumb} \rangle$  evidence a (qualitative) exponential increase with respect to  $n$ , as reported in Fig. 3.8, top right. As reference, the precise values of the evaluated mean running and tumbling time intervals are given in Table 3.6, together with their ratio, for three values of  $n$ . The running-to-tumbling ratio, which decreases as  $n$  increases, highlights the relevance of the number of flagella and the necessity of their synchronization with respect to the chemotactic behavior of the bacterium. That is, we see that for  $n = 1$  the time spent in running or tumbling motions is approximatively equivalent for the chosen  $\Delta t_{sim}$ , but if coordination among many flagella ( $n = 10$ ) has to take place, then the running motions are highly reduced with respect to tumbling motions, which is in agreement with biological expectations.

The results for  $\langle \Delta t_{adapt} \rangle$  are reported in Fig. 3.8, bottom, and in Table 3.6. In this case, it is not possible to recognize a simple function for the curve progress, and we see that the variation of the time intervals is within a range of a few tens of seconds. This result is biologically plausible, as the response of the bacterium to an environmental change (i.e., the addition or removal of ligands) should not be strictly dependent on the number of flagella that are present on its surface, otherwise the chemotactic pathway would not guarantee an appropriate adaptation mechanism to all bacteria, independently from the variation of the number of flagella that distinct cells can have.

The mean running and tumbling time intervals that we have evaluated for  $n = 1$  are in line with previous experimental observations [40, 41]. Anyway, to the best of our knowledge, no measurements of the synchronization time of multiple flagella have been reported in literature. As a matter of fact, CW and CCW times are typically measured for *single* flagellar motor in tethered bacteria, that is, in living and motile cells immobilized on a surface by attaching a single filament to the substratum and then letting the whole cell body spin. This experimental setup is unlikely applicable to simultaneously measure the CW and CCW times of *all* flagella occurring on a bacterium surface, therefore our results should be considered as model predictions that require the design of novel laboratory protocols for their validation.



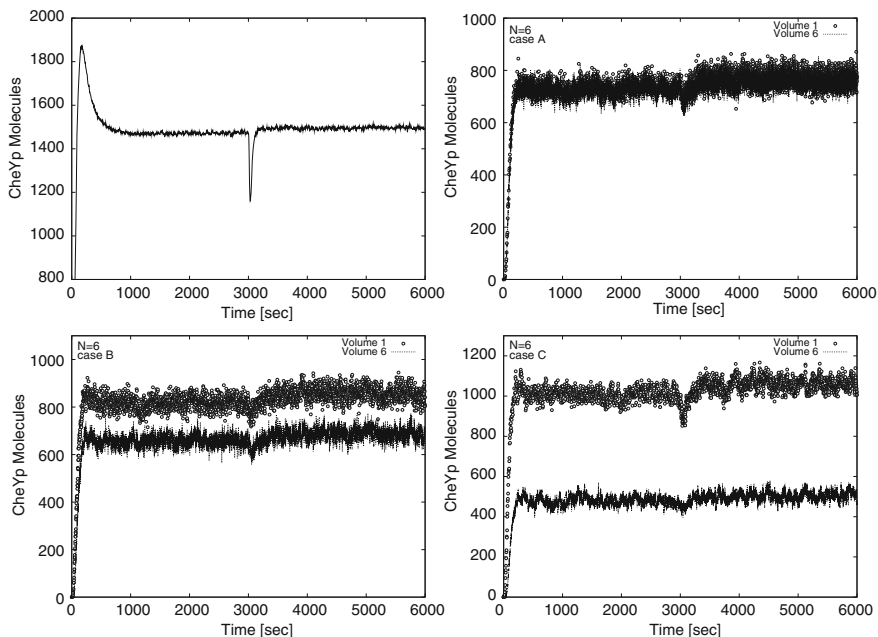
**Fig. 3.8** Variation of mean time values of running motions (*top left*), tumbling motions (*top right*), and adaptation time (*bottom*), with respect to the number of flagella

### 3.4.3 Simulations of $\Upsilon_{MV}$

In this section we present the results of multivolume stochastic simulations for the dynamics of CheYp. We discuss, in particular, the effects on the pathway response of distinct diffusion constants and of different numbers  $N$  of virtual volumes in which the bacterial volume is partitioned (where  $N = 6, 12, 30, 60$ ).

In Fig. 3.9 we show the results of stochastic simulations of the dynamics of CheYp, obtained with the single volume model described by  $\Upsilon_{SV}$  where no diffusive events occur (top left), and with the multivolume model  $\Upsilon_{MV}$  with  $N = 6$  virtual volumes obtained by using different diffusive reaction constants (cases A, B, C in the other three graphics, as described in Sect. 3.3.3). For the sake of readability, for each of the three cases A, B and C of the multivolume model, we picture only the dynamics of CheYp occurring inside  $V_1$  (the virtual volume containing the receptor complexes) and  $V_6$  (the virtual volume at the distal end of the cell), being the other internal volumes characterized by intermediate behaviors between these two.

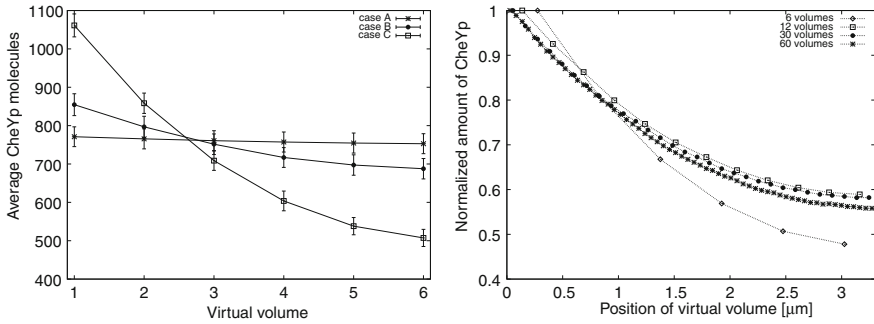
If we compare the results of  $\Upsilon_{MV}$  with those of  $\Upsilon_{SV}$ , the graphics clearly show that: (1) in all tested cases, the initial increase in CheYp amount of the single volume model is no more observable in the multivolume model, because of the effect of



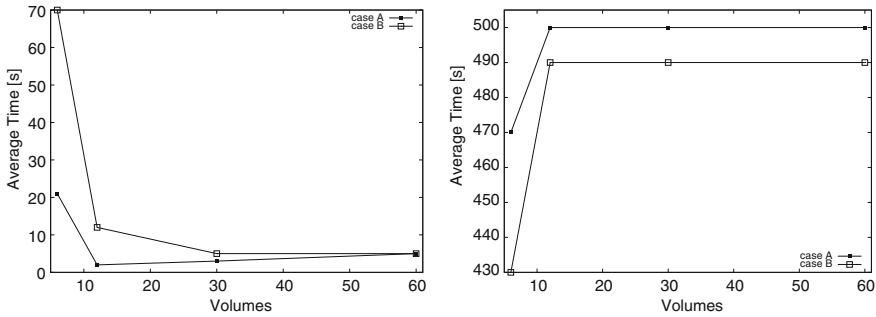
**Fig. 3.9** Comparison between the dynamics of CheYp before and after ligand addition (at time  $t = 3,000$  s), in the single volume model  $\gamma_{SV}$  (top left) and in the  $N = 6$  virtual volumes model  $\gamma_{MV}$ , for the cases A (top right), B (bottom left), C (bottom right)

diffusion that does not allow CheYp to accumulate but, on the contrary, immediately intervenes to distribute the molecules throughout the cell; (2) in all tested cases, the system is always able to reach a steady state value, and also to return to the pre-stimulus level of CheYp after the sensing of external ligand (simulated by the addition of ligand molecules to multiset  $M_1$  at time  $t = 3,000$  s); (3) the negative peak in CheYp amount due to ligand addition is much more pronounced in the single volume model with respect to each of the three cases of the multivolume model. Once more, this response is due to the diffusion process, which induces the presence of fewer molecules of CheYp inside each virtual volume with respect to the amount occurring in the single volume model, and therefore causes a slighter decrease of CheYp. Indeed, we can see that the most evident response after ligand addition occurs in case C, where the reduced diffusion constants allow to have, inside volume  $V_1$ , an amount of CheYp that is higher than the corresponding amounts of cases A and B inside the same volume. On the other side, inside volume  $V_6$  of cases A and B we can still see a small decrease of CheYp after ligand addition, which cannot be individuated in case C, as the slow diffusion flattens the dynamics.

In Fig. 3.10, left side, we show the gradients of CheYp obtained in the three cases of diffusion. Stochastic simulations have evidenced that, as long as the value of  $N$  increases, the molecular amounts of CheYp inside each virtual volume decrease,



**Fig. 3.10** *Left* gradients of CheYp in the multivolume model  $\gamma_{MV}$  with  $N = 6$  in cases A, B, C (vertical bars represent the width of stochastic fluctuations). *Right* comparison of gradients for  $\gamma_{MV}$  with  $N = 6, 12, 30, 60$  in case C



**Fig. 3.11** Average times of running (*left*) and tumbling motions (*right*), in cases A and B, considering a uniform distribution of seven flagella over the cellular surface

therefore inducing wider stochastic fluctuations of CheYp and reducing its negative peak after ligand addition (data not shown). As an example, we show in Fig. 3.10, right side, the gradients of CheYp obtained in case C for  $N = 6, 12, 30, 60$  (vertical bars representing the width of stochastic fluctuations have not been reported in this graphic for the sake of readability). Moreover, we can see that for each  $N$  the gradients of CheYp present qualitatively similar behaviors.

The simulation outcomes of the multivolume model are in line with previous experimental measurements and *in silico* investigations. Spatial gradients in the concentration of CheYp were observed using FRET imaging methods in single *E. coli* cells, showing that the highest concentration is located close to the receptor cluster [42]. Simulated concentration gradients of CheYp were previously shown in [10, 35] under different conditions of diffusion and localization of protein CheZ.

Finally, in Fig. 3.11, we show a comparison of running (left side) and tumbling (right side) average time in cases A and B, under the assumption that 7 flagella are uniformly distributed over the cellular surface. Running and tumbling intervals have been evaluated over a time interval of 500 s at steady state, using the procedure

given in Sect. 3.4.2; for each  $N$ , the threshold has been fixed as the mean (over all subvolumes) of the mean value of CheYp inside each subvolume. The results of Fig. 3.11 highlight how the values of stochastic constants associated to diffusive reactions affect the running and tumbling average time; in particular, by reducing the values of diffusion constants, the different gradient of CheYp occurring within the bacterium leads to higher running times. This effect is due to the fact that volumes placed at the distal end of the cell usually contain an amount of CheYp that is always below the threshold considered in the synchronization procedure.

### 3.5 Conclusion

In this work we have presented an application of membrane systems to the modeling, simulation and analysis of a signal transduction pathway. In particular, the modeling features of P systems have been exploited to develop an accurate mechanistic model of bacterial chemotaxis;  $\tau$ -DPP, a class of membrane systems, has been used to perform stochastic simulations of the dynamics of this model and to analyze its properties. The SBML file of the single volume model, together with simulation results, are available at BioSimWare website [43]. The descriptive power of  $\tau$ -DPP has been employed to develop a multivolume version of the model of bacterial chemotaxis in order to investigate the diffusion processes occurring within the cell.

Concerning the analysis of the single volume model of bacterial chemotaxis, we have investigated the possible influence of stochastic fluctuations of the chemotactic protein CheYp on the running motion of bacterial cells, with respect to an increasing number of flagella in the individual bacterium. To this aim, we have defined a procedure to identify the synchronization of CCW rotations of all flagella, and then we have compared the mean time intervals of running and tumbling motions of the cell, as well as of adaptation times to ligand addition, according to the different numbers of flagella. On the one hand, we have shown that the running-to-tumbling ratio highlights the relevance of the number of flagella, and the necessity of their synchronization with respect to the chemotactic behavior of the bacterium. On the other hand, the adaptation time does not seem to be strongly influenced by the varying number of flagella in distinct individual cells. These results have been obtained by performing stochastic simulations of a very detailed mechanistic model of the bacterial chemotaxis pathway, that takes into account all proteins, and their respective interactions, involved in both signaling and response. All post-translational modifications of proteins, such as methylation and phosphorylation, have been explicitly taken into account because of their relevant roles in the feedback control mechanisms governing this pathway. In particular, by exploiting  $\tau$ -DPP, we have investigated the dynamics of the pivotal protein involved in chemotaxis, CheYp, under different conditions, such as the deletion of other chemotactic proteins, the addition of distinct amounts of external ligand, the effect of different methylation states of the receptors.

Furthermore, we have extended the single volume model of bacterial chemotaxis starting from the consideration that computational investigations aimed at capturing

the features of spatial heterogeneity at the molecular level, due to the diffusion of chemicals through the cellular space, necessarily require an approach able to deal with distinct spatial domains, as well as to track the time evolution of interacting molecules within different locations. Following this approach and exploiting the modeling and simulation features of  $\tau$ -DPP, we have split the bacterial volume into adjacent and non overlapping “slices”, modeled by communicating virtual volumes, which allow to analyze the diffusion of proteins CheYp, CheY and CheZ throughout the whole volume. Simulations of the multivolume model have shown that the diffusion of molecules can induce larger stochastic fluctuations, thus altering the global response of this transduction pathway. In particular, diffusion events give rise to the formation of a cytoplasmic gradient of protein CheYp. However, we defer to a future work the investigation of quantitative aspects related to diffusion events, which is required to measure how the different diffusion regimes induce smaller or larger stochastic fluctuations and hence are possibly able to interfere with the response and the functional activity of bacterial flagella. In fact, the flagellar motors that are distributed into distinct virtual volumes are exposed to a local amount of CheYp that is generally different from the amount occurring in the neighboring virtual volumes.

In conclusion, we believe that the definition of detailed mechanistic models, like those proposed in this chapter for chemotaxis, coupled with the use of efficient procedures for the analysis of stochastic processes in individual cells, can be a good benchmark to investigate the combined roles of many biological factors interplaying within a common system. With this perspective, the development of formal methods specifically devised for the analysis of properties (e.g., synchronization) of stochastic systems represents indeed a hot research topic in biological modeling. In this context, the results that we have achieved with the single and multivolume models of chemotaxis represent novel hypotheses on the functioning of flagella coordination mechanisms in bacterial cells, which take place under the effect of diffusive events in the cytoplasm, stochastic fluctuations of CheYp and the distribution of flagellar motors over the cell surface. Now, it would be surely interesting to verify the occurrence and the influence of all these molecular phenomena in living and motile cells, by designing the necessary experimental protocols and hence feeding back to Biology our model predictions.

**Acknowledgments** Giancarlo Mauri and Dario Pescini acknowledge the partial funding by Regione Lombardia, research project “Network Enabled Drug Design (NEDD)”.

## References

1. G. Ciobanu, G. Păun, M.J. Pérez-Jiménez (eds.), *Applications of Membrane Computing* (Springer, Berlin, 2005)
2. G. Păun, G. Rozenberg, A. Salomaa, *The Oxford Handbook of Membrane Computing* (Oxford University Press, Oxford, 2010)
3. G. Păun, *Membrane Computing. An Introduction* (Springer, Berlin, 2002)
4. P systems web page. <http://ppage.psystems.eu/>



5. M.S. Jurica, B.L. Stoddard, Mind your B's and R's: bacterial chemotaxis, signal transduction and protein recognition. *Structure* (London, England : 1993), **6**(7), 809–813 (1998)
6. U. Alon, M.G. Surette, N. Barkai, S. Leibler, Robustness in bacterial chemotaxis. *Nature* **397**(6715), 168–171 (1999)
7. P.A. Spiro, J.S. Parkinson, H.G. Othmer, A model of excitation and adaptation in bacterial chemotaxis. *PNAS* **94**(14), 7263–7268 (1997)
8. C.J. Morton-Firth, D. Bray, Predicting temporal fluctuations in an intracellular signalling pathway. *J. Theor. Biol.* **192**(1), 117–128 (1998)
9. M.D. Levin, C.J. Morton-Firth, W.N. Abouhamad, R.B. Bourret, D. Bray, Origins of individual swimming behavior in bacteria. *Biophys. J.* **74**(1), 175–181 (1998)
10. K. Lipkow, S.S. Andrews, D. Bray, Simulated diffusion of phosphorylated CheY through the cytoplasm of *Escherichia coli*. *J. Bacteriol.* **187**(1), 45–53 (2005)
11. C.V. Rao, J.R. Kirby, A.P. Arkin, Design and diversity in bacterial chemotaxis: a comparative study in *Escherichia coli* and *Bacillus subtilis*. *PLoS Biol.* **2**(2), E49 (2004)
12. E. Mosca, P. Cazzaniga, D. Pescini, G. Mauri, L. Milanesi, Modelling spatial heterogeneity and macromolecular crowding with membrane systems, in *Proceedings of CMC11-Eleventh International Conference on Membrane Computing*, ed. by M. Gheorghe, T. Hinze, G. Păun, G. Rozenberg, A. Salomaa, LNCS, Vol. 6501, pp. 285–304 (2011)
13. K. Takahashi, S. Arjunan, M. Tomita, Space in systems biology of signaling pathways-towards intracellular molecular crowding in silico. *FEBS Lett.* **579**(8), 1783–1788 (2005)
14. D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977)
15. F. Baras, M. Mansour, Reaction-diffusion master equation: a comparison with microscopic simulations. *Phys. Rev. E* **54**, 6139–6148 (1996)
16. D.T. Gillespie, General method for numerically simulating stochastic time evolution of coupled chemical-reactions. *J. Comput. Phys.* **22**, 403–434 (1976)
17. P. Cazzaniga, D. Pescini, D. Besozzi, G. Mauri, Tau leaping stochastic simulation method in P systems, in *7th International Workshop on Membrane Computing WMC 2006*, ed. by H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa, LNCS, vol. 4361, pp. 298–313 (2006)
18. G. Păun, Computing with membranes. *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000)
19. Y. Cao, D.T. Gillespie, L.R. Petzold, Efficient step size selection for the tau-leaping simulation method. *Chin. J. Chem. Phys.* **124**(4), 044109 (2006)
20. P. Cazzaniga, D. Pescini, D. Besozzi, G. Mauri, S. Colombo, E. Martegani, Modeling and stochastic simulation of the Ras/cAMP/PKA pathway in the yeast *Saccharomyces cerevisiae* evidences a key regulatory function for intracellular guanine nucleotides pools. *J. Biotechnol.* **133**(3), 377–385 (2008)
21. D. Pescini, D. Besozzi, C. Zandron, G. Mauri, Analysis and simulation of dynamics in probabilistic P systems, in *Proceedings of 11th International Workshop on DNA Computing, DNA11*, ed. by A. Carbone, N.A. Pierce, LNCS, vol. 3892 (2006), pp. 236–247
22. D. Pescini, D. Besozzi, G. Mauri, Investigating local evolutions in dynamical probabilistic P systems, in *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)* (IEEE Computer Press, New York, 2005), pp. 440–447
23. D. Pescini, D. Besozzi, G. Mauri, C. Zandron, Dynamical probabilistic P systems. *Int. J. Found. Comput. Sci.* **17**(1), 183–204 (2006)
24. A. Bren, M. Eisenbach, Changing the direction of flagellar rotation in bacteria by modulating the ratio between the rotational states of the switch protein FliM. *J. Mol. Biol.* **312**(4), 699–709 (2001)
25. G.H. Wadhams, J.P. Armitage, Making sense of it all: bacterial chemotaxis. *Nature Rev. Mol. Cell Bio.* **5**(12), 1024–1037 (2004)
26. D. Besozzi, P. Cazzaniga, M. Dugo, D. Pescini, G. Mauri, A study on the combined interplay between stochastic fluctuations and the number of flagella in bacterial chemotaxis. *EPTCS* **6**, 47–62 (2009)
27. T.S. Shimizu, D. Bray, Modelling the bacterial chemotaxis receptor complex. *Novartis Found. Symp.* **247**, 162–177 (2002)

28. V. Sourjik, Receptor clustering and signal processing in *E. coli* chemotaxis. *Trends Microbiol.* **12**(12), 569–576 (2004)
29. C.J. Morton-Firth, T.S. Shimizu, D. Bray, A free-energy-based stochastic simulation of the Tar receptor complex. *J. Mol. Biol.* **286**(4), 1059–1074 (1999)
30. G. Li, R.M. Weis, Covalent modification regulates ligand binding to receptor complexes in the chemosensory system of *Escherichia coli*. *Cell* **100**(3), 357–365 (2000)
31. B.A. Mello, Y. Tu, Perfect and near-perfect adaptation in a model of bacterial chemotaxis. *Biophys. J.* **84**(5), 2943–2956 (2003)
32. D. Besozzi, P. Cazzaniga, S. Cocolo, G. Mauri, D. Pescini, Modeling diffusion in a signal transduction pathway: the use of virtual volumes in P systems. *Int. J. Found. Comput. Sci.* **22**(01), 89–96 (2011)
33. S. Schulmeister, M. Ruttorf, S. Thiem, D. Kentner, D. Lebiecz, V. Sourjik, Protein exchange dynamics at chemoreceptor clusters in *Escherichia coli*. *PNAS* **105**(17), 6403–6408 (2008)
34. K. Lipkow, Changing cellular location of CheZ predicted by molecular simulations. *PLoS Comput. Biol.* **2**(4), e39 (2006)
35. K. Lipkow, D. Odde, Model for protein concentration gradients in the cytoplasm. *Cell. Mol. Bioeng.* **1**(1), 84–92 (2008)
36. D. Bernstein, Simulating mesoscopic reaction-diffusion systems using the Gillespie algorithm. *Phys. Rev. E* **71**(4), 041103 (2005)
37. N. Barkai, S. Leibler, Robustness in simple biochemical networks. *Nature* **387**(6636), 913–917 (1997)
38. C. Kim, M. Jackson, R. Lux, S. Khan, Determinants of chemotactic signal amplification in *Escherichia coli*. *J. Mol. Biol.* **307**, 119–135 (2001)
39. B.E. Scharf, K.A. Fahrner, L. Turner, H.C. Berg, Control of direction of flagellar rotation in bacterial chemotaxis. *PNAS* **95**(1), 201–206 (1998)
40. M. Eisenbach, A. Wolf, M. Welch, S.R. Caplan, I.R. Lapidus, R.M. Macnab, H. Aloni, O. Asher, Pausing, switching and speed fluctuation of the bacterial flagellar motor and their relation to motility and chemotaxis. *J. Mol. Biol.* **211**(3), 551–563 (1990)
41. E. Korobkova, T. Emonet, J.M. Vilar, T.S. Shimizu, P. Cluzel, From molecular noise to behavioural variability in a single bacterium. *Nature* **428**(6982), 574–578 (2004)
42. A. Vaknin, H.C. Berg, Single-cell FRET imaging of phosphatase activity in the *Escherichia coli* chemotaxis system. *PNAS* **101**(49), 17072–17077 (2004)
43. Biosimware web page. <http://biosimware.disco.unimib.it>

## Chapter 4

# Membrane System-Based Models for Specifying Dynamical Population Systems

M. A. Colomer-Cugat, M. García-Quismondo, L. F. Macías-Ramos,  
M. A. Martínez-del-Amor, I. Pérez-Hurtado, M. J. Pérez-Jiménez,  
A. Riscos-Núñez and L. Valencia-Cabrera

**Abstract** Population Dynamics P systems (PDP systems, in short) provide a new formal bio-inspired modelling framework, which has been successfully used for modelling population dynamics on real ecosystems. The semantics of these systems is captured by the Direct distribution based on Consistent Blocks Algorithm (DCBA), which has been engineered into software simulation tools. In particular, *MeCoSim* (Membrane Computing Simulator) is a GUI developed in the framework of *P-Lingua* that can be used as a simulation environment for running virtual experiments. The parameters of each scenario to be simulated can be easily adjusted in a visual way, as well as the settings for the desired output format, thus facilitating the validation

---

M. A. Colomer-Cugat (✉)

Department of Mathematics, University of Lleida, Avinguda Alcalde Rovira Roure,  
191, 25198 Lleida, Spain  
e-mail: colomer@matematica.udl.cat

M. García-Quismondo · L. F. Macías-Ramos · M. A. Martínez-del-Amor · I. Pérez-Hurtado  
M. J. Pérez-Jiménez · A. Riscos-Núñez · L. Valencia-Cabrera  
Research Group on Natural Computing Department of Computer Science and Artificial  
Intelligence, University of Sevilla, Avenida Reina Mercedes s/n, 41012 Sevilla, Spain  
e-mail: mgarciaquismondo@us.es

L. F. Macías-Ramos  
e-mail: lfmaciasr@us.es

M. A. Martínez-del-Amor  
e-mail: mdelamor@us.es

I. Pérez-Hurtado  
e-mail: perezh@us.es

M. J. Pérez-Jiménez  
e-mail: marper@us.es

A. Riscos-Núñez  
e-mail: ariscosn@us.es

L. Valencia-Cabrera  
e-mail: lvalencia@us.es

of the designed models against real data. The simulation of PDP systems is data intensive for large models. Therefore, the development of efficient simulators for this field is needed. In fact, the computational power of GPUs is currently being used to accelerate simulations of PDP systems. We illustrate the modelling framework presented with a case study concerning pandemics.

## 4.1 Introduction

Mathematical models are abstract representations of real-world complex systems onto a mathematical/computational domain. They use symbolic notations and formalisms rather than physical devices to represent and analyse the relationships which describe the system under study. Moreover, never do they contain all features of their real-world counterpart, as they would be the real-world complex system itself. Models highlight the relevant features while ignoring the irrelevant ones. Therefore, a mathematical model should be regarded as a description of our current knowledge about a phenomenon of interest, rather than an exact representation of the truth.

The use of models is inherent to any scientific activity. When examining a phenomenon, scientists regularly use abstractions of reality such as diagrams, graphs, laws, etc. with the aim of describing and understanding it. Designing a model for a biological system is intrinsically a complicated task, since there is usually a large number of important factors that need to be considered. Therefore, it is advisable to make efforts to minimize the number of parameters, as well as the interactions between them.

Nowadays, ordinary differential equations (ODEs) constitute the most widely used approach for the study of complex systems, and in particular for population dynamics. However, this approach has some drawbacks, since it has been reported that the deterministic and continuous approach followed by ODEs is questionable in some complex systems. Consequently, in recent years new models based on the latest computational paradigms and technological advances have been adopted. Some examples are Petri nets [1], process algebra ( $\pi$ -calculus [2], bioambients [3], brane calculus [4],  $\kappa$ -calculus [5], etc.), state charts [6], agent based systems [7], and viability models [8, 9]. Although each of these computational frameworks captures some aspects regarding complex systems and their components, none of them fully integrates their dynamics and structural details.

Membrane Computing is an emergent branch of Natural Computing introduced by Păun in [10], inspired by the structure and functioning of living cells. This research field was introduced with the purpose of defining unconventional distributed and parallel computing devices, called P systems. The key differential feature of such systems is the so-called *membrane structure*, which represents the compartmentalization in the structural organization of cells. One of the main advantages of P systems, regarding their potential use as a modelling framework, is that their elements are more similar to those used in population dynamics than the abstractions of other formalisms. In this work we show how to capture, in a natural way, some features relevant to populations by using membrane structures.

We present P systems as a high level computational modelling framework which integrates the structural and dynamical aspects of complex systems in a comprehensive and relevant way while providing the required formalisation to perform mathematical and computational analysis. Rather than an alternative to more classical modelling frameworks, such as ODEs, P systems constitute a complementary approach to be used in those cases where the previous approaches fail. Among the most important properties of these models one can include their capacity to work in parallel, as well as capture randomness. P systems explicitly represent the discrete character of the quantity of components of a system by using rewriting rules on multisets. Objects on these multisets represent relevant parameters, as well as individuals. Although each complex system has its own important peculiarities, the vast majority of them share some common aspects. Some of these aspects are: (a) large number of individuals; (b) life cycle consisting of some basic processes; (c) periodic repetitions; (d) the evolution often depends on the environment; and (e) the natural dynamics suffers modifications due to human activities. These common features impose some computational requirements on the models. Among them, one can shortlist the following: many processes take place simultaneously, there exists cooperation between different individuals and elements, there exists partial synchronization among the dynamic evolution sub-problems, and some stages are cyclically repeated.

These considerations led to the definition of a special-purpose type of P systems. In particular, *Population Dynamics P system* models have been designed to study the evolution of the habitats corresponding to three real case studies: the bearded vulture (*Gypaetus barbatus*) in the Catalan Pyrenees (Spain) [11, 12], the zebra mussel (*Dreissena polymorpha*) at the reservoir of Ribarroja (Spain) [13] and, recently, the Pyrenean brook newt (*Calotriton asper*) in the river Segre (*Serra del Cadí*, Pyrenees) [14]. In the first case, the purpose of the obtained model is to study the evolution of the considered ecosystem under different scenarios in order to make the most appropriate management decisions for the conservation of an endangered species. The second case study corresponds to a completely different situation: *Dreissena polymorpha* is an exotic invasive species that has displayed an excellent adaptation following its introduction in the reservoir. Its uncontrolled reproduction causes significant economic and ecological losses. Hence, the goal in this case is to learn how to minimize the mussel population at the reservoir. In the latter case, the experts have documented repeatedly dramatic population losses of the studied species caused by severe floods. The goal is to evaluate whether there exists actual risk for the species to become extinct in this area as a consequence of extreme rainfall. In all three cases we have designed a simulator to validate the models. Actually, three different tools have been released, enabling the corresponding users to perform virtual experiments under different conditions.

This chapter is structured as follows: Sect. 4.2 is devoted to define concepts and notations that we will use throughout the text. A P systems based probabilistic modelling framework is considered in Sect. 4.3, and an inference engine capturing the semantics of the model is presented in Sect. 4.4. Section 4.5 is devoted to the software tools implementing the theoretical framework. Section 4.6 describes the modelling of a case study about an outbreak of a pandemic disease, in order to illustrate the

theoretical and practical framework presented. To conclude the chapter, in Sect. 4.7 we present some final considerations and outline some future work on the topic.

## 4.2 Preliminaries

In this section we introduce some concepts and notations which we will use throughout this chapter.

An *alphabet*,  $\Gamma$ , is a finite non-empty set whose elements are called *symbols*. A *multiset*,  $w$ , over an alphabet,  $\Gamma$ , is a pair  $(\Gamma, f)$  where  $f : \Gamma \rightarrow \mathbb{N}$  is a mapping. If  $w = (\Gamma, f)$  is a multiset then its *support* is defined as  $\text{supp}(w) = \{x \in \Gamma \mid f(x) > 0\}$ . We denote  $x \in w$  when  $x \in \text{supp}(w)$  and  $x^n \in w$  when  $x \in w$  and  $f(x) = n$ . A multiset is finite if its support is a finite set. If  $w = (\Gamma, f)$  is a finite multiset over  $\Gamma$ , and  $\text{supp}(w) = \{a_1, \dots, a_k\}$  then it will be denoted as  $w = a_1^{f(a_1)} \dots a_k^{f(a_k)}$  (here the order is irrelevant), and we say that  $f(a_1) + \dots + f(a_k)$  is the cardinality of  $w$ , denoted by  $|w|$ . The empty multiset is denoted by  $\emptyset$ . We also denote by  $M_f(\Gamma)$  the set of all finite multisets over  $\Gamma$ .

If  $w_1 = (\Gamma, f_1)$ ,  $w_2 = (\Gamma, f_2)$  are multisets over  $\Gamma$ , then we define the union of  $w_1$  and  $w_2$  as  $w_1 + w_2 = (\Gamma, g)$ , where  $g = f_1 + f_2$ .

In what follows, we assume the reader is already familiar with the basic notions and the terminology of P systems. For details, see [15].

## 4.3 A P Systems-Based Probabilistic Modelling Framework

*Population Dynamics P systems* are a variant of *multienvironment P systems with active membranes* [16], a model with a network of environments, each of them containing a P system with features such as electrical charges associated with membranes to describe specific properties in a better way. All P systems share the same *skeleton*, in the sense that they have the same working alphabet, the same membrane structure and the same set of rules. Nevertheless, the probability functions associated with the rules can vary among environments, and also the initial multisets are independent.

In what follows, the formal definition of PDP systems is introduced. To continue, some additional definitions are given regarding to the rules of the systems. These definitions will be specially useful within the scope of the Direct distribution based on Consistent Blocks Algorithm (DCBA), an inference engine for PDP systems that will be covered in Sect. 4.4. Finally, some desirable properties of the PDP systems model are reviewed at the end of this section.

### 4.3.1 PDP Systems

**Definition 1** A Population Dynamics P system (PDP) of degree  $(q, m)$ ,  $q, m \geq 1$ , taking  $T \geq 1$  time units, is a tuple

$$\Pi = (G, \Gamma, \Sigma, T, \mathcal{R}_E, \mu, \mathcal{R}, \{f_{r,j} \mid r \in \mathcal{R}, 1 \leq j \leq m\}, \{\mathcal{M}_{ij} \mid 1 \leq i \leq q, 1 \leq j \leq m\})$$

where:

- $G = (V, S)$  is a directed graph with  $V = \{e_1, \dots, e_m\}$ .
- $\Gamma$  and  $\Sigma$  are alphabets such that  $\Sigma \subsetneq \Gamma$ .
- $T$  is a natural number.
- $\mathcal{R}_E$  is a finite set of communication rules of the form  $(x)_{e_j} \xrightarrow{pr} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ , where  $x, y_1, \dots, y_h \in \Sigma$ ,  $(e_j, e_{j_l}) \in S$  for all  $1 \leq l \leq h$ , and  $p_r : \{1, \dots, T\} \rightarrow [0, 1]$  is a computable function such that for each  $e_j \in V$  and  $x \in \Sigma$ , the sum of functions associated with the rules of the type  $(x)_{e_j} \xrightarrow{pr} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$  is the constant function 1.
- $\mu$  is a rooted tree injectively labelled by  $1 \leq i \leq q$ , and by symbols from the set  $\{0, +, -\}$ .
- $\mathcal{R}$  is a finite set of evolution rules of the form  $u[v]_i^\alpha \rightarrow u'[v']_i^{\alpha'}$ , where  $u, v, u', v' \in M_f(\Gamma)$ ,  $u + v \neq \emptyset$ ,  $1 \leq i \leq q$  and  $\alpha, \alpha' \in \{0, +, -\}$ , such that there is no rules  $(x)_{e_j} \xrightarrow{pr} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$  and  $u[v]_i^\alpha \rightarrow u'[v']_i^{\alpha'}$  having  $x \in u$ .
- For each  $r \in \mathcal{R}$  and  $1 \leq j \leq m$ ,  $f_{r,j} : \{1, \dots, T\} \rightarrow [0, 1]$  is a computable function such that for each  $u, v \in M_f(\Gamma)$ ,  $1 \leq i \leq q$ ,  $\alpha, \alpha' \in \{0, +, -\}$  and  $1 \leq j \leq m$ , the sum of functions  $f_{r,j}$  with  $r \equiv u[v]_i^\alpha \rightarrow u'[v']_i^{\alpha'}$ , is the constant function 1.
- For each  $i, j$  ( $1 \leq i \leq q, 1 \leq j \leq m$ ),  $\mathcal{M}_{ij} \in M_f(\Gamma)$ .

A Population Dynamics P system defined as above can be viewed as a set of  $m$  environments  $e_1, \dots, e_m$  interlinked by the edges from the directed graph  $G$ . Each environment  $e_j$  can only contain symbols from alphabet  $\Sigma$  and all of them also contain a P system skeleton,  $\Pi_j = (\Gamma, \mu, \mathcal{M}_{1,j}, \dots, \mathcal{M}_{q,j}, \mathcal{R})$ , of degree  $q$ , where:

- (a)  $\Gamma$  is the working alphabet whose symbols are also called objects.
- (b)  $\mu$  is a rooted tree which describes a membrane structure consisting of  $q$  membranes (nodes of the tree) injectively labelled by  $1, \dots, q$ . The skin membrane (the root of the tree) is labelled by 1 and its parent is the environment  $e_j$ . We also associate one with each membrane.
- (c)  $\mathcal{M}_{1,j}, \dots, \mathcal{M}_{q,j}$  are finite multisets over  $\Gamma$ , describing the objects initially associated to the  $q$  membranes of  $\mu$ , within the environment  $e_j$ .
- (d)  $\mathcal{R}$  is the set of evolution rules of each P system. Every rule  $r \in \mathcal{R}$  in  $\Pi_j$  has a computable function  $f_{r,j}$  associated with it. For each environment  $e_j$ , we denote by  $\mathcal{R}_{\Pi_j}$  the set of rules with probabilities obtained by coupling each  $r \in \mathcal{R}$  with the corresponding function  $f_{r,j}$ .

Furthermore, there is a set  $\mathcal{R}_E$  of communication rules between environments, and the natural number  $T$  represents the simulation time of the system. The set of rules of the whole system is  $\bigcup_{j=1}^m \mathcal{R}_{\Pi_j} \cup \mathcal{R}_E$ .

The *semantics* of Population Dynamics P systems is defined through a non deterministic and synchronous model, in the sense that a global clock is assumed. Next, we describe some semantic aspects of these systems.

A communication rule  $r \in \mathcal{R}_E$ , of the form  $(x)_{e_j} \xrightarrow{pr} (y_1)_{e_{j_1}} \dots (y_h)_{e_{j_h}}$  is applicable to environment  $e_j$  if it contains object  $x$ . When such a rule is applied, object  $x$  passes from  $e_j$  to  $e_{j_1}, \dots, e_{j_h}$  possibly transformed into objects  $y_1, \dots, y_h$  respectively. At any moment  $t$  ( $1 \leq t \leq T$ ) for each object  $x$  in environment  $e_j$ , if there exist communication rules of type  $(x)_{e_j} \xrightarrow{pr} (y_1)_{e_{j_1}} \dots (y_h)_{e_{j_h}}$ , then one of these rules will be applied. If more than one such a rule can be applied to an object at a given instant, the system selects one randomly, according to their associated functions.

In each  $\Pi_j$ , an evolution rule  $r \in \mathcal{R}$ , of the form  $u[v]_i^\alpha \rightarrow u'[v']_i^{\alpha'}$ , is applicable to membrane  $i$ , whose electrical charge is  $\alpha$ , and that contains multiset  $v$ , and whose parent contains multiset  $u$ . When such a rule is applied, the objects of the multisets  $v$  and  $u$  are removed from membrane  $i$  and from its parent membrane, respectively. Simultaneously, objects in multiset  $u'$  are introduced into the parent of membrane  $i$ , and objects of multiset  $v'$  are introduced into membrane  $i$ . The application also replaces the charge of membrane  $i$  with  $\alpha'$ . In each environment  $e_j$ , the rule  $r$  has associated a probability function  $f_{r,j}$  that provides an index of the applicability when several rules compete for objects.

For each  $j$  ( $1 \leq j \leq m$ ) there is just one further restriction, concerning the consistency of charges: in order to apply several rules of  $\mathcal{R}_{\Pi_j}$  simultaneously to the same membrane, all the rules must have the same electrical charge on their right-hand side.

An *instantaneous description* or *configuration* of the system at any instant  $t$  is a tuple of multisets specifying the objects associated to each environment and membrane, with its polarization, of every  $\Pi_j$ . We assume that all environments are initially empty and that all membranes have initially a neutral polarization. We assume a global clock, synchronizing all membranes and the application of all the rules (from  $\mathcal{R}_E$  and from  $\mathcal{R}_{\Pi_j}$  in all environments).

In each time unit a given configuration can be transformed into another by using rules from the whole system as follows: the rules to be applied are selected in a non-deterministic way according to the probabilities assigned to them, and all applicable rules are simultaneously applied in a maximal way. In this way, we get *transitions* from one configuration of the system to the next one.

A *computation* is a sequence of configurations such that the first term of the sequence is the initial configuration of the system, and each non-initial configuration is obtained from the previous one by applying rules of the system in a maximally parallel manner with the restrictions previously mentioned.



### 4.3.2 Additional Definitions

Next, we define some concepts associated with the rules from the system that will be used in the Sect. 4.3.3.

**Definition 2** Given a rule  $r \in \mathcal{R}$  of the form  $u[v]_i^\alpha \rightarrow u'[v']_i^{\alpha'}$  where  $1 \leq i \leq q$ ,  $\alpha, \alpha' \in \{0, +, -\}$  and  $u, v, u', v' \in M_f(\Gamma)$ :

- The left-hand side of  $r$  is  $LHS(r) = (i, \alpha, u, v)$ . The charge of  $LHS(r)$  is  $charge(LHS(r)) = \alpha$ .
- The right-hand side of  $r$  is  $RHS(r) = (i, \alpha', u', v')$ . The charge of  $RHS(r)$  is  $charge(RHS(r)) = \alpha'$ .

**Definition 3** Given a rule  $r \in \mathcal{R}_E$  of the form  $(x)_{e_j} \xrightarrow{Pr} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ , the left-hand side of  $r$  is  $LHS(r) = (e_j, x)$ , and the right-hand side of  $r$  is  $RHS(r) = (e_{j_1}, y_1) \cdots (e_{j_h}, y_h)$ .

**Definition 4** A block of rules is a set of rules with the same left-hand side. We say that a block of rules is consistent if any pair of its rules can be applied in a simultaneous manner provided that there are enough objects.

Rules from  $\mathcal{R}_E$  can be classified into blocks in a natural way.

**Definition 5** Given  $(e_j, x)$ ,  $1 \leq j \leq m$ , and  $x \in \Sigma$ , the block associated with  $(e_j, x)$  is the set:

$$B_{e_j, x} = \{r \in \mathcal{R}_E \mid LHS(r) = (e_j, x)\}$$

Extending left-hand side definition,  $LHS(B_{e_j, x}) = (e_j, x)$ .

Bearing in mind that the restriction concerning the consistency of charges only applies to membranes, but not to environments, any block of rules from  $\mathcal{R}_E$  is obviously a consistent block.

Let us then focus on the case of evolution rules from  $\mathcal{R}$ . Such rules can be classified into *consistent blocks* according to the following definition.

**Definition 6** Given  $(i, \alpha, \alpha', u, v)$ ,  $1 \leq i \leq q$ ,  $\alpha, \alpha' \in \{0, +, -\}$ , and  $u, v \in M_f(\Gamma)$ , the consistent block associated with  $(i, \alpha, \alpha', u, v)$  is the set:

$$B_{i, \alpha, \alpha', u, v} = \{r \in \mathcal{R} \mid LHS(r) = (i, \alpha, u, v) \wedge charge(RHS(r)) = \alpha'\}$$

Extending left-hand side definition,  $LHS(B_{i, \alpha, \alpha', u, v}) = (i, \alpha, u, v)$  and the charge is  $\alpha$ .

That is, the consistent block associated with  $(i, \alpha, \alpha', u, v)$  is the set of rules  $r \in \mathcal{R}$  whose left-hand side is  $(i, \alpha, u, v)$  and such that the electrical charge of the membrane after their execution is the same:  $\alpha'$ . Therefore, any pair of rules from  $B_{i, \alpha, \alpha', u, v}$  can be simultaneously applied provided that there are enough objects. In that case, membrane  $i$  will modify its polarization from  $\alpha$  to  $\alpha'$ .

We recall that, according to the semantics of our model, the sum of probabilities of all the rules belonging to the same block is always equal to 1; in particular, rules whose probability is equal to 1 form individual blocks. Note that rules with overlapping (but different) left-hand sides are classified into different blocks. The latter leads to object *competition*, what is a critical aspect to manage with the simulation algorithms. This suggests introducing the concept of mutual consistency among blocks.

**Definition 7** Two blocks  $B_{i_1, \alpha_1, \alpha'_1, u_1, v_1}$  and  $B_{i_2, \alpha_2, \alpha'_2, u_2, v_2}$  are mutually consistent with each other, if and only if  $(i_1 = i_2 \wedge \alpha_1 = \alpha_2) \Rightarrow (\alpha'_1 = \alpha'_2)$ .

That is, for two blocks mutually consistent with each other and with the same charge any rule of the first block can be simultaneously applied together with any rule of the second block, provided that there are enough objects.

Since rules in  $\mathcal{R}_E$  (communication between environments) do not affect the electrical charge of any membrane and they do not interfere at all with the applicability of the rest of the rules, a block  $B_{e_{j1}, x_1}$  is always mutually consistent with any other  $B_{e_{j2}, x_2}$ , as well as with any block  $B_{i, \alpha, \alpha', u, v}$  of evolution rules.

**Definition 8** Given  $x \in \Gamma$ ,  $l \in H$ , and  $r \in \mathcal{R}$  such that  $LHS(r) = (i, \alpha, u, v)$ , we say that  $(x, l)$  appears in  $LHS(r)$  with multiplicity  $k$  in any of the following cases:

- $l = i$ , and  $x$  appears in multiset  $v$  with multiplicity  $k$
- $l$  is the label of the parent of membrane  $i$ , and  $x$  appears in multiset  $u$  with multiplicity  $k$

### 4.3.3 Some Properties of PDP Systems Models

The following four properties [2] are desirable for any computational model:

- **Relevance:** A computational model must capture some essential features of the system investigated.

It should present a unifying specification of its physical structure and the different components that constitute the system, the interaction between them and their dynamical behaviour.

PDP systems are able to successfully capture the relevance of the underlying system by associating objects to individuals or other significant elements, and the interaction between them by means of evolution rules.

- *Computability*: It should be possible to implement or simulate a model in a computer, so that one can run simulations to study the dynamics of the system by manipulating experimental conditions in the model. In this manner, the model can be experimentally validated, and moreover the behaviour of the system under different scenarios of interest can be studied. The computability of the model also allows us to perform model checking and similar techniques to infer and to study qualitative and quantitative properties of the system in an automatic way. In this respect, the model should be mathematically tractable. That is, it should be possible to perform mathematical analysis on it.

Since P systems are computing devices, the computability of PDP systems is an inherent property. Moreover, the inference engine (as detailed in Sect. 4.4) captures the semantics of the studied dynamics models.

- *Understandability*: The abstract formalism used should correspond well to the informal concepts and ideas which are used by the experts in the population under study.

As PDP systems objects and rules capture in a simple manner the behaviour and dynamical interaction of the relevant elements in the modelled system, they are easy to be understood by experts in the problem domain.

- *Extensibility*: It should be easy to identify the different components and characteristics of the systems that are essential in the context of the management or scientific problem to be solved or comprehended [17], so they can be rearranged, duplicated, composed, etc. in an easy way to produce other models. Models of complex systems should also be extensible to higher levels of organizations. PDP system rule design is module-oriented, favouring low coupling between them. This approach allows a mostly independent modules development while enabling the addition, removal and/or reuse of them to the system. As modules are designed in a separate way, different levels of complexity can be achieved in each one.

## 4.4 An Inference Engine: The DCBA Algorithm

Within the framework of Membrane Computing the goal of a simulation algorithm is to select and execute, for each time unit, an applicable multiset of rules. The Direct distribution based on Consistent Blocks Algorithm (DCBA) follows this approach, paying special attention to the proportional distribution of objects among competing blocks (with overlapping LHS), thus determining the number of times that each rule in  $\bigcup_{j=1}^m \mathcal{R}_{\Pi_j} \cup \mathcal{R}_E$  is applied. See [18] for a more detailed explanation and examples of how this algorithm works.

Algorithm 1 describes the main loop of the DCBA. It follows the same general scheme as its predecessors, DNDP and BBB [19] where the simulation of a transition step is structured in two stages: selection and execution. The firststage (selection)

selects which rules are to be applied (and how many times) on each environment. The second stage (execution) implements the effects of applying the previously selected rules, yielding the next configuration of the PDP system. Note that, although every  $\Pi_j$  shares the same set of rules  $R$ , the probability functions may differ for each environment.

As shown in Algorithm 1, the selection stage consists of three phases: Phase 1 distributes objects to the blocks in a proportional way, as it will be explained later on; Phase 2 assures the *maximality* by assigning to some of the blocks the objects still unassigned after Phase 1; and finally, Phase 3 translates block applications into rule applications by computing random numbers following the multinomial distribution with the corresponding *probabilities*.

---

### Algorithm 1 DCBA MAIN PROCEDURE

---

**Require:** A Population Dynamics P system of degree  $(q, m)$ ,  $T \geq 1$ , and  $A \geq 1$

```

1:  $C_0 \leftarrow$  Initial configuration
2: INITIALIZATION ▷ (Algorithm 2)
3: for  $t \leftarrow 1$  to  $T$  do
4: Calculate probability functions  $f_{r,j}(t)$  and  $p_r(t)$ 
5:  $C'_t \leftarrow C_{t-1}$ 
6: SELECTION of rules:
    - PHASE 1: Distribution ▷ (Algorithm 3)
    - PHASE 2: Maximality ▷ (Algorithm 4)
    - PHASE 3: Probabilities ▷ (Algorithm 5)
7: EXECUTION of rules. ▷ (Algorithm 6)
8:  $C_t \leftarrow C'_t$ 
9: end for

```

---

The *INITIALIZATION* procedure (Algorithm 2) constructs a static distribution table  $\mathcal{T}_j$  for each environment. Two variables,  $B_{sel}^j$  and  $R_{sel}^j$ , are also initialized, in order to store the selected multisets of blocks and rules, respectively.

**Observation 1** *Each column label of the tables  $\mathcal{T}_j$  contains the information of the corresponding block left-hand side.*

**Observation 2** *Each row of the tables  $\mathcal{T}_j$  contains the information related to the object competitions: for a given object, its row indicates which blocks are competing for it (those columns having non-null values).*

**Algorithm 2** INITIALIZATION

---

1: Construction of the *static distribution* table  $\mathcal{T}$ :

- Column labels: consistent blocks  $B_{i,\alpha,\alpha',u,v}$  of rules from  $R$ .
- Row labels: pairs  $(o, i)$  and  $(x, 0)$ , for all objects  $o \in \Gamma$ ,  $x \in \Sigma$  and membrane label  $i$ , being 0 the identifier of the environment.
- For each cell of the table: place  $\frac{1}{k}$  if its row label  $(o, i)$  appears with multiplicity  $k > 0$  in the LHS of its column label  $B_{i,\alpha,\alpha',u,v}$ .

- 2: **for**  $j = 1$  **to**  $m$  **do**  $\triangleright$  (Construct the *static expanded* tables  $\mathcal{T}_j$ )
  - 3:  $\mathcal{T}_j \leftarrow \mathcal{T}$ .  $\triangleright$  (Initialize the table with the original  $\mathcal{T}$ )
  - 4: For each rule block  $B_{e_j,x}$  from  $\mathcal{R}_E$ , add a column labelled by  $B_{e_j,x}$  to the table  $\mathcal{T}_j$ ; place the value 1 at row  $(x, 0)$  for that column
  - 5: Initialize the multisets  $B_{sel}^j \leftarrow \emptyset$  and  $R_{sel}^j \leftarrow \emptyset$
  - 6: **end for**
- 

The distribution of objects among the blocks with overlapping LHS (competing blocks) is performed in selection Phase 1 (Algorithm 3). The expanded static tables  $\mathcal{T}_j$  are used for this purpose in each environment, together with three different filter procedures. FILTER 1 discards the columns of the table corresponding to non-applicable blocks due to mismatch charges in the LHS and in the configuration  $C'_t$ . Then, FILTER 2 discards the columns with objects in the LHS not appearing in  $C'_t$ . Finally, in order to save space in the table, FILTER 3 discards empty rows. These three filters are applied at the beginning of Phase 1, and the result is a *dynamic table*  $\mathcal{T}_j^t$  (for the environment  $j$  and time step  $t$ ).

The semantics of the modelling framework requires a set of mutually consistent blocks before distributing objects to the blocks. For this reason, after applying FILTERS 1 and 2, the mutual consistency is checked. Note that this checking can be easily implemented by a loop over the blocks. If it fails, meaning that an inconsistency was encountered, the simulation process is halted, providing a warning message to the user. Nevertheless, it could be interesting to find a way to continue the execution by non-deterministically constructing a subset of mutually consistent blocks. Since this method can be exponentially expensive in time, it is optional for the user whether to activate it or not.

Once the columns of the *dynamic table*  $\mathcal{T}_j^t$  represent a set of mutually consistent blocks, the distribution process starts. This is carried out by creating a temporal copy of  $\mathcal{T}_j^t$ , called  $\mathcal{TV}_j^t$ , which stores the following products:

- The normalized value with respect to the row: this is a way to *proportionally* distribute the corresponding object along the blocks. Since it depends on the multiplicities in the LHS of the blocks, the distribution, in fact, penalizes the blocks requiring more copies of the same object. This is inspired in the amount of energy required to gather individuals from the same species.
- The value in the dynamic table (i.e.  $\frac{1}{k}$ ): this indicates the number of possible applications of the block with the corresponding object.
- The multiplicity of the object in the configuration  $C'_t$ : this performs the distribution of the number of copies of the object along the blocks.

**Algorithm 3** SELECTION PHASE 1: DISTRIBUTION

---

```

1: for  $j = 1$  to  $m$  do ▷ (For each environment  $e_j$ )
2:   Apply filters to table  $\mathcal{T}_j$  using  $C'_t$ , obtaining  $\mathcal{T}_j^t$ , as follows:
      a.  $\mathcal{T}_j^t \leftarrow \mathcal{T}_j$ 
      b. FILTER 1 ( $\mathcal{T}_j^t, C'_t$ )
      c. FILTER 2 ( $\mathcal{T}_j^t, C'_t$ )
      d. Check mutual consistency for the blocks remaining in  $\mathcal{T}_j^t$ . If there is at least one
         inconsistency then report the information about the error, and optionally halt the
         execution (in case of not activating step 3)
      e. FILTER 3 ( $\mathcal{T}_j^t, C'_t$ )
3:   (OPTIONAL) Generate a set  $S_j^t$  of sub-tables from  $\mathcal{T}_j^t$ , formed by sets of
      mutually consistent blocks, in a maximal way in  $\mathcal{T}_j^t$  (by the inclusion
      relationship). Replace  $\mathcal{T}_j^t$  with a randomly selected table from  $S_j^t$ .
4:    $a \leftarrow 1$ 
5:   repeat
6:     for all rows  $X$  in  $\mathcal{T}_j^t$  do
7:        $RowSum_{X,t,j} \leftarrow$  total sum of the non-null values in the row  $X$ 
8:     end for
9:      $\mathcal{TV}_j^t \leftarrow \mathcal{T}_j^t$  ▷ (A temporal copy of the dynamic table)
10:    for all non-null positions  $(X, Y)$  in  $\mathcal{T}_j^t$  do
11:       $mult_{X,t,j} \leftarrow$  multiplicity in  $C'_t$  at  $e_j$  of the object at row  $X$ 
12:       $\mathcal{TV}_j^t(X, Y) \leftarrow \lfloor mult_{X,t,j} \cdot \frac{(\mathcal{T}_j^t(X,Y))^2}{RowSum_{X,t,j}} \rfloor$ 
13:    end for
14:    for all not filtered column, labelled by block  $B$ , in  $\mathcal{T}_j^t$  do
15:       $N_B \leftarrow \min_{X \in rows(\mathcal{T}_j^t)} (\mathcal{TV}_j^t(X, B))$  ▷ (The minimum of the column)
16:       $B_{sel}^j \leftarrow B_{sel}^j + \{B^{N_B}\}$  ▷ (Accumulate the value to the total)
17:       $C'_t \leftarrow C'_t - LHS(B) \cdot N_B$  ▷ (Delete the LHS of the block)
18:    end for
19:    FILTER 2 ( $\mathcal{T}_j^t, C'_t$ )
20:    FILTER 3 ( $\mathcal{T}_j^t, C'_t$ )
21:     $a \leftarrow a + 1$ 
22:  until  $(a > A) \vee$  (all the selected minimums at step 15 are 0)
23: end for

```

---

**Algorithm 4** SELECTION PHASE 2: MAXIMALITY

---

```

1: for  $j = 1$  to  $m$  do ▷ (For each environment  $e_j$ )
2:   Set a random order to the blocks remaining in the last updated table  $\mathcal{T}_j^t$ 
3:   for all block  $B$ , following the previous random order do
4:      $N_B \leftarrow$  number of possible applications of  $B$  in  $C'_t$ 
5:      $B_{sel}^j \leftarrow B_{sel}^j + \{B^{N_B}\}$  ▷ (Accumulate the value to the total)
6:      $C'_t \leftarrow C'_t - LHS(B) \cdot N_B$  ▷ (Delete the LHS of block  $B$ ,  $N_B$  times)
7:   end for
8: end for

```

---

After the object distribution process, the number of applications for each block is calculated by selecting the minimum value in each column. This number is then used for consuming the LHS from the configuration. However, this application could be non-maximal. The distribution process can eventually deliver objects to blocks that are restricted by other objects. As this situation may occur frequently, the distribution and the configuration update process is performed  $A$  times, where  $A$  is an input parameter referring to *accuracy*. The more the process is repeated, the more accurate the distribution becomes at the expense of simulation performance. We have experimentally checked that  $A = 2$  gives the best accuracy/performance ratio. In order to efficiently repeat the loop for  $A$ , and also before going to the next phase (maximality), it is interesting to apply FILTERS 2 and 3 again.

After Phase 1, it may be the case that some blocks are still applicable to the remaining objects. This may be caused by a low  $A$  value or by rounding artefacts in the distribution process. Due to the requirements of P systems semantics, a maximality phase is now applied (Algorithm 4). Following a random order, a maximal number of applications is calculated for each block which is still applicable.

After the application of Phases 1 and 2, a maximal multiset of selected (mutually consistent) blocks has been computed. The output of the selection stage has to be, however, a maximal multiset of selected rules. Hence, Phase 3 (Algorithm 5) passes from blocks to rules, by applying the corresponding probabilities (at the local level of blocks). The rules belonging to a block are selected according to a multinomial distribution  $M(N, g_1, \dots, g_l)$ , where  $N$  is the number of applications of the block, and  $g_1, \dots, g_l$  are the probabilities associated with the rules  $r_1, \dots, r_l$  within the block, respectively.

---

**Algorithm 5** SELECTION PHASE 3: PROBABILITY
 

---

```

1: for  $j = 1$  to  $m$  do ▷ (For each environment  $e_j$ )
2:   for all block  $B^{N_B} \in B_{sel}^j$  do
3:     Calculate  $\{n_1, \dots, n_l\}$ , a random multinomial  $M(N_B, g_1, \dots, g_l)$  with
       respect to the probabilities of the rules  $r_1, \dots, r_l$  within the block.
4:     for  $k = 1$  to  $l$  do
5:        $R_{sel}^j \leftarrow R_{sel}^j + \{r_k^{n_k}\}$ .
6:     end for
7:   end for
8:   Delete the multiset of selected blocks  $B_{sel}^j \leftarrow \emptyset$ . ▷ (Useful for the next step)
9: end for
  
```

---

Finally, the execution stage (Algorithm 6) is applied. This stage consists on adding the RHS of the previously selected multiset of rules, as the objects present on the LHS of these rules have already been consumed. Moreover, the indicated membrane charge is set.

**Algorithm 6 EXECUTION**


---

```

1: for  $j = 1$  to  $m$  do                                     ▷ (For each environment  $e_j$ )
2:   for all rule  $r^n \in R_{sel}^j$  do                               ▷ (Apply the RHS of selected rules)
3:      $C'_t \leftarrow C'_t + n \cdot RHS(r)$ 
4:     Update the electrical charges of  $C'_t$  from  $RHS(r)$ .
5:   end for
6:   Delete the multiset of selected rules  $R_{sel}^j \leftarrow \emptyset$ .    ▷ (Useful for the next step)
7: end for

```

---

## 4.5 Simulation of PDP Systems

As already mentioned in Sect. 4.1, computational models (and, in particular, P system-based models) are assumed to rely on software simulators that carry out virtual experiments in order to evaluate the usefulness of the formal model defined. At the current stage, multiple software tools have been developed in the Membrane Computing field (see e.g. [20]), most of them specifically tailored for a single type of P systems.

This section describes a set of software tools providing the needed infrastructure to define, simulate and virtually experiment with PDP systems.

### 4.5.1 P-Lingua, and the pLinguaCore Library

Each P system model features characteristic semantic constraints that determine not only the type of rules allowed, but also the way in which rules are applied. This general information is embedded in the simulator engine, but in order to perform simulations, we need additional information regarding the P system to be simulated. The term *simulator input* will be used to refer to this initial data which provides the formal specification of the P system.

One possible approach to implement the simulator input could be to require a specific input file for each simulator, or directly insert the data into the source code. This approach imposes a specific format for the input, given by the data structures used in the design of each software simulator. Moreover, if we wanted to run many different experiments, a great redundant effort would be required. An alternative approach could be to standardize the simulator input by establishing a common format. These two approaches raise a trade-off. On the one hand, specific simulator inputs could be defined in a more straightforward way, as the used format is closer to the P system features to simulate. On the other hand, although the latter approach involves analysing different P system models to develop a standard format, there is no need to develop completely a new simulator every time a new P system should be simulated, as it is possible to use a common software library in order to parse the standard input format. Moreover, users would no longer be obliged to learn a

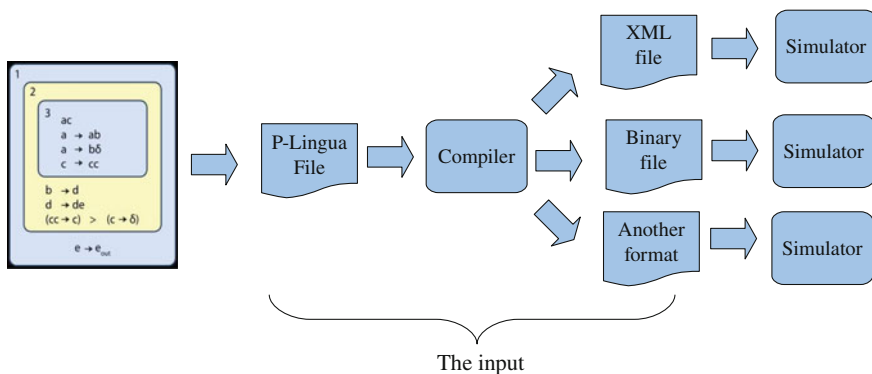


new input format every time they use a different simulator, and they would not need to rewrite the specification of P systems which are going to be simulated every time they move on to another model. This second approach is the one considered in P-Lingua [21, 22], a specification language to define P systems within several P system models.

The P-Lingua project also provides free software tools under GNU GPL license [23] for compilation, simulation and debugging tasks. The main tools are integrated in a Java library called *pLinguaCore*. These tools include a parser to handle P-Lingua input files and check possible programming errors (both lexical/syntax and semantics). They also include several built-in sequential simulators to generate P system computations for the supported models. Furthermore, these tools can export the P-Lingua definition file into other file formats in order to get interoperability between different software environments. The approach to define the simulator input by using the P-Lingua framework is illustrated in Fig. 4.1 where we can see how P system definitions in P-Lingua format can be translated into other file formats by using *pLinguaCore*, eventually becoming the input for different simulator environments and gaining interoperability. Moreover, such input is free of programming errors, since the parser inside *pLinguaCore* has already checked them.

Since the initial release version, each new update of P-Lingua and *pLinguaCore* includes new supported models and implements new simulation algorithms, while in parallel fixing some bugs found. The current release (*pLinguaCore* 3.0, available for download at the P-Lingua website [22]), covers the following P system models:

- (Cell-like) Transition P systems.
- (Cell-like) Symport/antiport P systems.
- (Cell-like) Active membranes with division rules.
- (Cell-like) Active membranes with creation rules.
- (Cell-like) Probabilistic P systems.
- Tissue-like P systems with division rules.
- Population Dynamics P systems (PDP systems)



**Fig. 4.1** The P-Lingua approach to define simulators input

As mentioned above, pLinguaCore includes at least one built-in simulator for each supported model. In particular, a sequential implementation of the algorithm discussed in this chapter for PDP systems (DCBA) is included, along with some alternatives (e.g. DNDP).

P-Lingua and pLinguaCore 3.0 can be used to assist in the design of PDP systems. It can also be used as simulation core for other software tools such as MeCoSim, as it will be explained below.

### 4.5.2 The Visual Environment MeCoSim

The availability of a general language, P-Lingua, to define P systems, along with a set of tools to parse, simulate and debug models based on this kind of systems, enables the designer to work with P systems in a high level of abstraction. In this context, an abstract problem is defined in P-Lingua format, but a number of scenarios (instances of the problem) can be analysed and simulated for that abstract problem. As the variety and size of the problems to model and simulate with this framework increase, the need for visual tools for modelling and simulation arises—in such a way that a P-Lingua model for a family of problems can be specified by a designer user, while the information about each specific scenario can be included by an end user in a visual way.

This need leads to software developments for providing Graphical User Interfaces (GUI) to introduce the input data for a specific scenario, both for the description of the initial configuration and for some variable parameters for the model. Moreover, such GUIs also take care of rendering some outputs showing the required information, possibly including tables and charts. Some examples of these applications were successfully used to model and simulate ecosystems [11, 13, 24].

While P-Lingua framework provided a general mechanism to define, simulate and debug P systems, each family of problems implied the design, development and maintenance of different ad-hoc GUIs. MeCoSim [25, 26], Membrane Computing Simulator, arises to solve the need of developing *ad-hoc* applications, by providing a general solution for defining custom GUIs adapted for each problem. The user can define a custom structure of tabs, input and output tables, charts, graphs, etc. for each given family of problems, so MeCoSim can be viewed as a new layer above P-Lingua framework, complementing its functionality.

The initial goal has been extended to provide a general integrated environment to work with P systems, including functionalities for modelling, simulation, debugging, analysis, properties extraction and verification of models based on P systems. MeCoSim platform provides a plugins architecture to extend the initial functionality with external programs.

### 4.5.3 Accelerating PDP Systems Simulations

Previous sections have introduced a simulation algorithm, DCBA, along with a sequential implementation inside pLinguaCore. However, the simulations were relatively slow, since pLinguaCore library is not performance-oriented.

In order to overcome this limitation, a more efficient implementation based on C++ and OpenMP was presented in [27], taking advantage of modern multicore architectures (e.g. 4-core Intel I5 and I7 microprocessors). These simulators save on memory by avoiding the creation of the static distribution table required in Phase 1. This feature (called *virtual table* solution) is carried out by translating the operations over the table to operations directly to rule blocks information. Only the row sums and the column minimums are stored in auxiliary data structure together with flags denoting that a column has been filtered. Concerning the parallelism implementation, simulations and environments were distributed along the cores. Runtime gains of up to  $2.5\times$  were achieved, so these preliminary results indicate that the simulation of PDP systems are memory bound.

Furthermore, since GPU computing [28] has been successfully used to implement other P systems simulators [29–31], the simulation of PDP systems on this technology was a natural step. NVIDIA's *CUDA* (Compute Unified Device Architecture) [32, 33] provides developers with a high-level programming model that allows them to take advantage of the NVIDIA's GPU parallel architecture. Most recent NVIDIA's GPUs (with Kepler architecture) provide thousands of cores, and a fast memory access. However, programs must fit data parallelism to achieve best performance.

The CUDA-based simulator for PDP systems [34] distributes simulations and environments along the multiprocessors of the GPU, and the rule blocks are parallelized within each multiprocessor. It has been benchmarked against a set of randomly generated PDP systems (without biological meaning), achieving speedups of  $7\times$  for large sizes (PDP systems with 50,000 rule blocks, 20 environments and running 50 simulations) on a NVIDIA Tesla C1060 GPU (240 cores and 4 GBytes of memory) in comparison to the multi-core CPU version. The source code is available under GPLv3 license, within the PMCGPU project [35], codenamed as *ABCD-GPU*.

## 4.6 A Case Study: Pandemics

A *pandemic* is an outbreak of a disease that occurs over a wide geographic area and affects an exceptionally high proportion of the population.

In this section, we present a SIR computational model based on Population Dynamics P systems. SIR is an acronym: **S** stands for the *susceptible population*, those who are not yet infected, but may become infected; **I** stands for the *infected population*, those who are ill and can transmit the disease, and **R** stands for the dead or recovered individuals that are removed from the infected population and cannot transmit the disease. The SIR mathematical model for pandemics is an ODEs based

model that has been used to understand the spatial-temporal transmission dynamics of influenza. This model refers to systems where no human interaction is considered (e.g. vaccination campaigns, declarations of quarantine, prophylactic actions, etc.).

Our case study will be restricted to three physically separated communities (e.g. in different cities). Each community is formed by four neighbourhoods, where basic facilities for daily life are available: schools, work places, shops, etc. Individuals in a neighbourhood are organized in families (families may have different structure or number of members). In addition, seven groups will be considered, according to their age: Daycares/Playgroups, Elementary schools, Middle schools, High schools, and two groups for Adults (19–53 years old, and over 53 years old). A susceptible person can be infected either in the bosom of the family, at work, or in leisure time (at the neighbourhood or when travelling). Figure 4.2 displays the network corresponding to breeding grounds for each age group.

4.6.1 Design of a PDP Modelling Pandemics

In this section, the model for the exposed case study by using PDP systems is presented. The model is composed by seven modules of rules. The first one of them (initial infection) will only be executed when the model is initialized choosing randomly infected people within the population. The execution of the remaining six

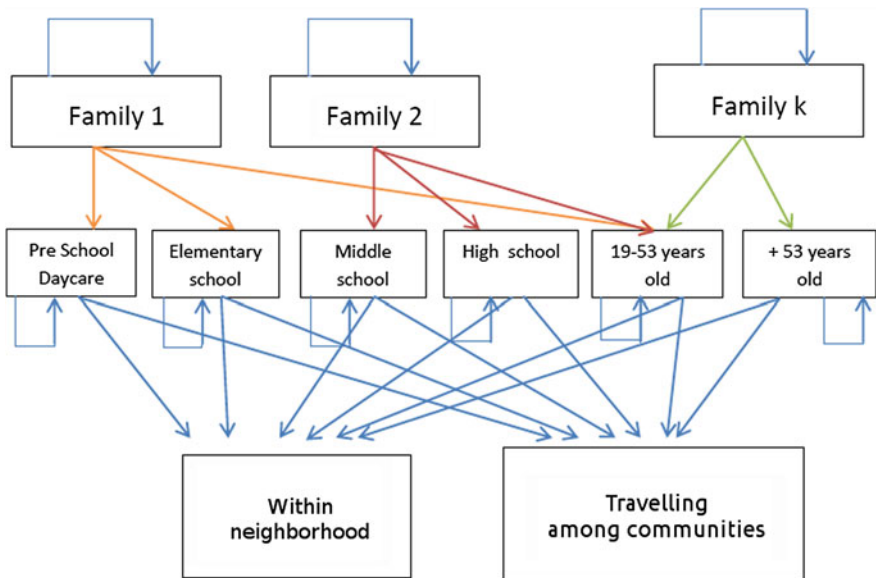


Fig. 4.2 A network of the infection flow

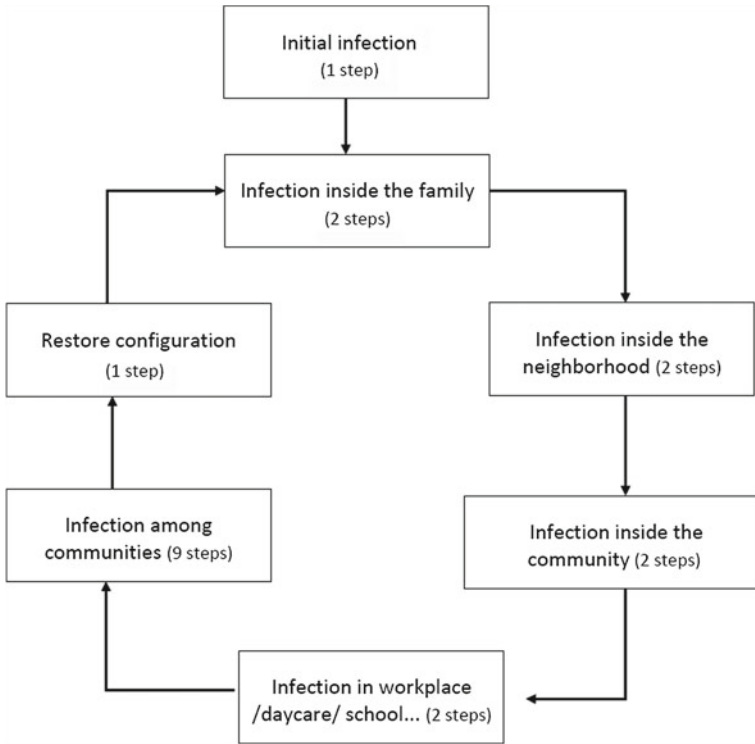
modules will be interpreted as the evolution of the pandemic scenario during one day.

The model consists of a PDP system of degree  $(2, 3)$  taking  $T \geq 1$  time units,

$$\Pi = (G, \Gamma, \Sigma, T, \mathcal{R}_E, \mu, \mathcal{R}, \{f_{r,j} \mid r \in \mathcal{R}, 1 \leq j \leq 3\}, \{\mathcal{M}_{1,j}, \mathcal{M}_{2,j}, 1 \leq j \leq 3\})$$

where:

- $G = (V, S)$  is a complete directed graph, with  $V = \{e_1, e_2, e_3\}$  and  $S = \{(e_i, e_j) \mid 1 \leq i, j \leq 3\}$  (Fig.4.3 shows the graph, including the P system inside each environment).



**Fig. 4.3** Modules corresponding to the SIR model

- The working alphabet,  $\Gamma$  is the set  
 $\{a_n \mid 1 \leq n \leq 4\} \cup \{C_i \mid 0 \leq i \leq 18\} \cup$   
 $\{X_{f,g}, \bar{X}_{f,g}, Y_{f,g}, Z_{f,g}, V_{f,g}, A_{f,g}, S_{f,g},$   
 $R_{f,g} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7\} \cup$   
 $\{\bar{A}_{f,g,i}, \bar{S}_{f,g,i} \mid$   
 $1 \leq f \leq 4F, 1 \leq g \leq 7, 2 \leq i \leq 5\} \cup$   
 $\{A'_{f,g,j}, \bar{X}'_{f,g,j}, X'_{f,g,j}, V'_{f,g,j}, W_{f,g,j}, W'_{f,g,j} \mid$   
 $1 \leq f \leq 4F, 1 \leq g \leq 7, 1 \leq j \leq 3\} \cup$   
 $\{\hat{A}, \bar{X}, \bar{Y}, \bar{Z}, \bar{V}, \bar{W}\} \cup \{\bar{M}'_j \mid 1 \leq j \leq 3\}$

Symbols  $a_n$  ( $1 \leq n \leq 4$ ) are used to represent in the initial configuration the individuals initially infected in each neighbourhood. The uninfected individuals are represented by symbols  $X_{f,g}$ ,  $X'_{f,g,j}$ ,  $Y_{f,g}$ ,  $Z_{f,g}$ ,  $V_{f,g}$ ,  $V'_{f,g,j}$ ,  $W_{f,g,j}$ ,  $W'_{f,g,j}$  (index  $f$  is associated with the families, index  $g$  is associated with the age groups, and index  $j$  is associated with the environments representing communities); the infected individuals are represented by symbols  $\bar{X}_{f,g}$  and  $\bar{X}'_{f,g,j}$ ; symbols  $\bar{S}_{f,g,i}$  and  $\bar{A}_{f,g,i}$  represent symptomatic and asymptomatic individuals, respectively (index  $i$  is associated with the stage of illness, indicating days since infection); symbols  $\bar{X}$ ,  $\bar{Y}$ ,  $\bar{Z}$ ,  $\bar{V}$ ,  $\bar{W}$  are used to model the interactions of individuals infected by others represented by symbols  $S_{f,g}$  and  $A_{f,g}$  (symptomatic and asymptomatic, respectively);  $\bar{V}'_j$ ,  $A'_{f,g,j}$  and  $\hat{A}$  are auxiliary symbols used for travelling among communities;  $R_{f,g}$  represent individuals who were infected but have been able to recover. A global clock,  $C_i$ , controls the evolution of the P system and the charge changes of membrane 2 along a cycle (that is, a day).

- $\Sigma = \{A'_{f,g,j}, V'_{f,g,j} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 1 \leq j \leq 3\} \cup$   
 $\{\bar{V}'_j \mid 1 \leq j \leq 3\} \cup$   
 $\{X'_{f,g,i,j} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 0 \leq i \leq 1, 1 \leq j \leq 3\}$
- $T = 18 \cdot \text{Days}$ , where  $\text{Days}$  is the number of days to simulate. Each day in the real scenario is simulated by 18 computational steps.
- $\mathcal{R}_E = \{r_{e1,f,g,j,i}, r_{e2,f,g,j,i} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 1 \leq i, j \leq 3\} \cup$   
 $\{r_{e3,j,i} \mid 1 \leq i, j \leq 3\} \cup$   
 $\{r_{e4,f,g,i,j,p} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 0 \leq i \leq 1, 1 \leq j, p \leq 3, j \neq p\} \cup$   
 $\{r_{e5,f,g,i,j} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 0 \leq i \leq 1, 1 \leq j \leq 3\}$
- $\mu = [\ ]_2$  is the membrane structure, and the corresponding initial multisets in the environment  $j$  are:
  - $\mathcal{M}_{1,j} = \emptyset$
  - $\mathcal{M}_{2,j} = \{X_{f,g}^{q_{f,g,j}} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 1 \leq j \leq 3\} \cup$   
 $\{a_n^{l_{n,j}} \mid 1 \leq n \leq 4, 1 \leq j \leq 3\} \cup \{C_0\}$

Objects  $X_{f,g}$  represent individuals in step 0, for families  $f$  from 1 to  $4F$ , with  $F$  the number of families per neighbourhood; they are divided by index  $g$  in 7 age groups as explained before. Objects  $a_n$ , as mentioned before, repre-

sent individuals initially infected at the beginning of the simulated period. The amount of objects depends, for each environment, on parameters  $q_{f,g,j}$  and  $l_{n,j}$ , that should be specified by the user.

- In what follows we enumerate the rules in  $\mathcal{R} \cup \mathcal{R}_E$  along with some comments on their functioning:

### Initial infection

These rules take care of generating the initial scenario, randomly distributing symptomatic and asymptomatic individuals along the system, according to the parameters  $l_{n,j}$ . These parameters are provided by the user before starting the simulation, and they indicate the initial amount of infected individuals on each environment  $e_j$  (represented by the multiplicity of objects  $a_n$ ).

These rules are only applied in the first step of the computation, they are not part of the loop that represents a day (see Fig. 4.4).

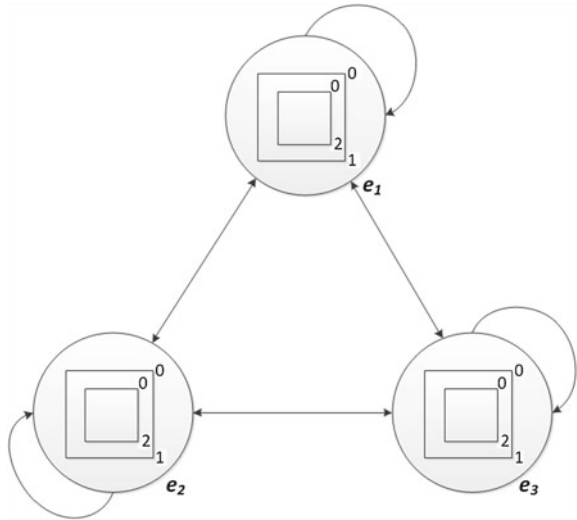
Each infected individual is estimated to interact with 20 other individuals during one day, and thus 20 copies of  $S_{f,g}$  ( $A_{f,g}$ , respectively) are generated for each symptomatic (asymptomatic, respectively) individual.

Generate symptomatic individuals

$$r_{1,f,g,i,n} \equiv [a_n X_{f,g} \xrightarrow{ps/4} \bar{S}_{f,g,i} S_{f,g}^{20}] 2 \begin{cases} (n-1)F < f \leq nF, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 5, \\ 1 \leq n \leq 4 \end{cases}$$

Generate asymptomatic individuals

**Fig. 4.4** Structure of the environment graph of the PDP system defined



$$r_{2,f,g,i,n} \equiv [a_n X_{f,g} \xrightarrow{(1-ps)/4} \bar{A}_{f,g,i} \tilde{X} A_{f,g}^{20}]_2 \begin{cases} (n-1)F < f \leq nF, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 5, \\ 1 \leq n \leq 4 \end{cases}$$

Clock advance

$$r_3 \equiv [C_0 \longrightarrow C_1]_2$$

### Infection inside the family (first step)

As we said before, any infected individual can spread the disease by interacting with other individuals. However, we distinguish two types of behaviour: symptomatic individuals are supposed to stay at home, hence only interacting with their relatives, while asymptomatic individuals are supposed to be unaware of their infection. In this module, rules  $r_{4,f,g,g'}$  and  $r_{5,f,g,g'}$  deal with the possible infection within the family. The rest of modules will cover different reasons for interaction between susceptible individuals and asymptomatic infected individuals.

The model also has special rules (see  $r_{6,f,g}$  and  $r_{7,f,g}$ ) to deal with the fact that two infected people, when interacting, cannot get infected again. Please note that these rules compete for objects  $A_{f,g}$  and  $S_{f,g}$  against rules  $r_{4,f,g,g'}$  and  $r_{5,f,g,g'}$ .

Infection of susceptible individuals

$$r_{4,f,g,g'} \equiv [S_{f,g} X_{f,g'}]_2 \longrightarrow [S_{f,g} \bar{X}_{f,g'} \tilde{X}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g, g' \leq 7 \end{cases}$$

$$r_{5,f,g,g'} \equiv [A_{f,g} X_{f,g'}]_2 \longrightarrow [A_{f,g} \bar{X}_{f,g'} \tilde{X}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g, g' \leq 7 \end{cases}$$

Interaction between infected individuals

$$r_{6,f,g} \equiv [S_{f,g} \tilde{X}]_2 \longrightarrow [S_{f,g} \tilde{X}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{7,f,g} \equiv [A_{f,g} \tilde{X}]_2 \longrightarrow [A_{f,g} \tilde{X}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

Clock advance with charge change

$$r_8 \equiv [C_1]_2 \longrightarrow [C_2]_2^-$$

### Infection inside the family (last step)

We need to avoid that rules for infection inside the family are applicable in more steps. Note that by changing the charge we guarantee that all symbols are renamed by means of the following rules:

Renaming of susceptible ( $X$ ) and infected ( $\tilde{X}$ ) individuals

$$r_{9,f,g} \equiv [X_{f,g}]_2^- \longrightarrow [Y_{f,g}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{10} \equiv [\tilde{X}]_2^- \longrightarrow [\tilde{Y}]_2$$



Removal of the possibility of infection by symptomatic individuals

$$r_{11,f,g} \equiv [S_{f,g}]_2^- \longrightarrow [\#]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

Clock advance with charge change

$$r_{12} \equiv [C_2]_2^- \longrightarrow [C_3]_2$$

### Infection inside the neighborhood (first step)

The following four blocks of rules represent the probability of infection by interacting with a “neighbour”. The set of  $4F$  families living in each community is divided into four intervals, representing four neighbourhoods. Thus, two individuals  $A_{f,g}$  and  $Y_{f',g'}$  live in the same neighbourhood if their family indexes ( $f$  and  $f'$ ) belong to the same interval.

This module includes, like discussed above, rules to deal with the interaction between two infected people (see  $r_{17,f,g}$ ).

Asymptomatic individuals affecting susceptible ones

$$\begin{aligned} r_{13,f,f',g,g'} &\equiv [A_{f,g} Y_{f',g'}]_2 \longrightarrow [A_{f,g} \bar{X}_{f',g'} \tilde{Y}]_2^- \left\{ \begin{array}{l} 1 \leq f, f' \leq F, \\ 1 \leq g, g' \leq 7 \end{array} \right. \\ r_{14,f,f',g,g'} &\equiv [A_{f,g} Y_{f',g'}]_2 \longrightarrow [A_{f,g} \bar{X}_{f',g'} \tilde{Y}]_2^- \left\{ \begin{array}{l} F < f, f' \leq 2F, \\ 1 \leq g, g' \leq 7 \end{array} \right. \\ r_{15,f,f',g,g'} &\equiv [A_{f,g} Y_{f',g'}]_2 \longrightarrow [A_{f,g} \bar{X}_{f',g'} \tilde{Y}]_2^- \left\{ \begin{array}{l} 2F < f, f' \leq 3F, \\ 1 \leq g, g' \leq 7 \end{array} \right. \\ r_{16,f,f',g,g'} &\equiv [A_{f,g} Y_{f',g'}]_2 \longrightarrow [A_{f,g} \bar{X}_{f',g'} \tilde{Y}]_2^- \left\{ \begin{array}{l} 3F < f, f' \leq 4F, \\ 1 \leq g, g' \leq 7 \end{array} \right. \end{aligned}$$

Interaction between infected individuals

$$r_{17,f,g} \equiv [A_{f,g} \tilde{Y}]_2 \longrightarrow [A_{f,g} \tilde{Y}]_2^- \left\{ \begin{array}{l} 3F < f, f' \leq 4F, \\ 1 \leq g, g' \leq 7 \end{array} \right.$$

Clock advance with charge change

$$r_{18} \equiv [C_3]_2 \longrightarrow [C_4]_2^-$$

### Infection inside the neighborhood (last step)

Following the same strategy as in the previous module (infection inside the family), we include now the corresponding renaming rules.

Renaming of susceptible ( $Y$ ) and infected ( $\tilde{Y}$ ) individuals

$$\begin{aligned} r_{19,f,g} &\equiv [Y_{f,g}]_2^- \longrightarrow [Z_{f,g}]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right. \\ r_{20} &\equiv [\tilde{Y}]_2^- \longrightarrow [\tilde{Z}]_2 \end{aligned}$$

Clock advance with charge change

$$r_{21} \equiv [C_4]_2^- \longrightarrow [C_5]_2$$

### Infection inside the community (first step)

This module of rules represents the probability of infection by interacting with an individual living in the same community. No conditions are imposed on the indexes of the symbols, if they are in the same membrane (within the same environment) then they can interact.

Rules to deal with the interaction between two infected people are included here as well (see  $r_{23,f,g}$ ).

Evolution of susceptible ( $Z$ ) and previously affected ( $\tilde{Z}$ ) individuals

$$r_{22,f,f',g,g'} \equiv [A_{f,g} Z_{f',g'}]_2 \longrightarrow [A_{f,g} \bar{X}_{f',g'} \tilde{Z}]_2^- \begin{cases} 1 \leq f, f' \leq 4F, \\ 1 \leq g, g' \leq 7 \end{cases}$$

$$r_{23,f,g} \equiv [A_{f,g} \tilde{Z}]_2 \longrightarrow [A_{f,g} \tilde{Z}]_2^- \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

Clock advance with charge change

$$r_{24} \equiv [C_5]_2 \longrightarrow [C_6]_2^-$$

### Infection inside the community (last step)

Following the same strategy as in the previous modules, we include now the corresponding renaming rules.

Renaming of susceptible ( $Z$ ) and infected ( $\tilde{Z}$ ) individuals

$$r_{25,f,g} \equiv [Z_{f,g}]_2^- \longrightarrow [V_{f,g}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{26} \equiv [\tilde{Z}]_2^- \longrightarrow [\tilde{V}]_2$$

Clock advance with charge change

$$r_{27} \equiv [C_6]_2^- \longrightarrow [C_7]_2$$

### Infection in workplace/daycare/school...

In this part of the day cycle, we consider the interactions which take place at work. More precisely, the individuals can now be infected by interacting with other individuals of the same age group (since they will both be studying at school, or both working, etc.).

The renaming step is not needed in this module. By giving a positive charge to membrane 2 (in the previous modules only 0 and  $-$  are used), the module for travelling among communities is initiated in the next step.

An asymptomatic individual may meet susceptible ones ( $V_{k,g}$ ), or another infected individual ( $\tilde{V}$ )

$$r_{28,f,f',g} \equiv [A_{f,g} V_{f',g}]_2 \longrightarrow [A_{f,g} \bar{X}_{f',g} \tilde{V}]_2^+ \left\{ \begin{array}{l} 1 \leq f, f' \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{29,f,g} \equiv [A_{f,g} \tilde{V}]_2 \longrightarrow [A_{f,g} \tilde{V}]_2^+ \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

Clock advance with charge change

$$r_{30} \equiv [C_7]_2 \longrightarrow [C_8]_2^+$$

### Infection among communities

This is the last possibility included in our model for spreading the disease. It considers the case of asymptomatic individuals traveling outside their communities.

In order to represent such “travels”, objects are sent out of membrane 2, and then out of membrane 1 into their environment. Then, communication rules are applied, possibly moving to a different environment, and after that the objects representing the travellers move into membrane 1 and into membrane 2, and then they are ready for the infection rules (see  $r_{52,f,g,j}$ ). Rule  $r_{53}$  has a double role: on one hand they are equivalent to rules for interaction between two infected people used in previous modules, but on the other hand, together with rules  $r_{55}$  and  $r_{56}$  they also take care of eliminating objects involved in infection rules, so that they will not interfere in the development of the next cycle.

Movement to skin membrane

$$r_{31,f,g} \equiv [V_{f,g}]_2^+ \longrightarrow V_{f,g} [ ]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{32} \equiv [\tilde{V}]_2^+ \longrightarrow \tilde{V} [ ]_2$$

$$r_{33,f,g} \equiv [A_{f,g}]_2^+ \longrightarrow A_{f,g} [ ]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{34,f,g} \equiv [\bar{X}_{f,g}]_2^+ \longrightarrow \bar{X}_{f,g} [ ]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{35,f,g,i} \equiv [\bar{A}_{f,g,i}]_2^+ \longrightarrow \bar{A}_{f,g,i} [ ]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 5 \end{array} \right.$$

$$r_{36,f,g,i} \equiv [\bar{S}_{f,g,i}]_2^+ \longrightarrow \bar{S}_{f,g,i} [ ]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 5 \end{array} \right.$$

$$r_{37,f,g} \equiv [R_{f,g}]_2^+ \longrightarrow R_{f,g} [ ]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

Clock advance with charge change

$$r_{38} \equiv [C_8]_2^+ \longrightarrow [C_9]_2$$

### Movement to the environment

$$r_{39,f,g,j} \equiv [A_{f,g}]_1 \xrightarrow{1/3} A'_{f,g,j} [ ]_1 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

$$r_{40,f,g,j} \equiv [V_{f,g}]_1 \xrightarrow{1/3} V'_{f,g,j} [ ]_1 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

$$r_{41,j} \equiv [\tilde{V}]_1 \xrightarrow{1/3} \tilde{V}'_j [ ]_1 \quad 1 \leq j \leq 3$$

### Clock advance

$$r_{42} \equiv [C_9 \longrightarrow C_{10}]_2$$

### Communication rules among environments

$$r_{e1,f,g,j,i} \equiv (A'_{f,g,i})_{e_j} \longrightarrow (\hat{A})_{e_i} \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq i, j \leq 3 \end{cases}$$

$$r_{e2,f,g,j,i} \equiv (V'_{f,g,i})_{e_j} \longrightarrow (W_{f,g,j})_{e_i} \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq i, j \leq 3 \end{cases}$$

$$r_{e3,j,i} \equiv (\tilde{V}'_i)_{e_j} \longrightarrow (\tilde{W})_{e_i} \quad 1 \leq i, j \leq 3$$

### Clock advance

$$r_{43} \equiv [C_{10} \longrightarrow C_{11}]_2$$

### Input into membrane 1

$$r_{44} \equiv \hat{A} [ ]_1 \longrightarrow [\hat{A}]_1$$

$$r_{45,f,g,j} \equiv W_{f,g,j} [ ]_1 \longrightarrow [W_{f,g,j}]_1 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

$$r_{46} \equiv \tilde{W} [ ]_1 \longrightarrow [\tilde{W}]_1$$

### Clock advance

$$r_{47} \equiv [C_{11} \longrightarrow C_{12}]_2$$

### Input into membrane 2

$$r_{48} \equiv \hat{A} [ ]_2 \longrightarrow [\hat{A}]_2$$

$$r_{49,f,g,j} \equiv W_{f,g,j} [ ]_2 \longrightarrow [W_{f,g,j}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

$$r_{50} \equiv \tilde{W} [ ]_2 \longrightarrow [\tilde{W}]_2$$

Clock advance

$$r_{51} \equiv [C_{12} \longrightarrow C_{13}]_2$$

An asymptomatic traveller may meet a susceptible individual ( $W_{f,g,j}$ ), or another infected individual ( $\tilde{W}$ )

$$r_{52,f,g,j} \equiv [\hat{A} W_{f,g,j}]_2 \longrightarrow [\bar{X}'_{f,g,j}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

$$r_{53} \equiv [\hat{A} \tilde{W}]_2 \longrightarrow [\#]_2^-$$

Clock advance with charge change

$$r_{54} \equiv [C_{13}]_2 \longrightarrow [C_{14}]_2^-$$

Start the reverse trip to return home (movement to skin membrane)

$$r_{55} \equiv [\hat{A}]_2^- \longrightarrow [\#]_2$$

$$r_{56} \equiv [\tilde{W}]_2^- \longrightarrow [\#]_2$$

$$r_{57,f,g,j} \equiv [W_{f,g,j}]_2^- \longrightarrow W'_{f,g,j} [ ]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

$$r_{58,f,g,j} \equiv [\bar{X}'_{f,g,j}]_2^- \longrightarrow \bar{X}'_{f,g,j} [ ]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

Clock advance with charge change

$$r_{59} \equiv [C_{14}]_2^- \longrightarrow [C_{15}]_2$$

Next step of the trip back (movement to the environment)

$$r_{60,f,g,j} \equiv [W'_{f,g,j}]_1 \longrightarrow X'_{f,g,j} [ ]_1 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

$$r_{61,f,g,j} \equiv [\bar{X}'_{f,g,j}]_1 \longrightarrow \bar{X}'_{f,g,j} [ ]_1 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j \leq 3 \end{cases}$$

Clock advance

$$r_{62} \equiv [C_{15} \longrightarrow C_{16}]_2$$

Next step of the trip back (communication rules among environments)

$$r_{e4,f,g,j,i} \equiv (X'_{f,g,j})_{e_i} \longrightarrow (X_{f,g})_{e_j} \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j, i \leq 3 \end{cases}$$

$$r_{e5,f,g,j,i} \equiv (\bar{X}'_{f,g,j})_{e_i} \longrightarrow (\bar{X}_{f,g})_{e_j} \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 1 \leq j, i \leq 3 \end{cases}$$

Clock advance

$$r_{63} \equiv [C_{16} \longrightarrow C_{17}]_2$$

Next step of the trip back (going into membrane 1)

$$r_{64,f,g} \equiv X_{f,g}[ ]_1 \longrightarrow [X_{f,g}]_1 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{65,f,g} \equiv \bar{X}_{f,g}[ ]_1 \longrightarrow [\bar{X}_{f,g}]_1 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

Clock advance with charge change

$$r_{66} \equiv [C_{17}]_2 \longrightarrow [C_{18}]_2^+$$

### Restore configuration

The last step of the “trip back” is slightly modified to be used as a restoring mechanism, so that a new day cycle can start (if the maximum number of steps  $T$  has not yet been reached).

There are three possibilities for individuals who have been marked as infected during this cycle: either they are considered not actually infected, even though they have been exposed (see  $r_{70,f,g}$ ); or they become infected and symptomatic (see  $r_{68,f,g}$ ); or they become infected and asymptomatic (see  $r_{69,f,g}$ ).

A third index is considered for symbols representing previously infected individuals. It represents the days since they became infected. We assume that after 5 days, asymptomatic individuals will change status to recovered. For symptomatic individuals, on the other hand, we have to take into account the probability of recovery, which is a parameter that may get different values for each age group.

Last step to complete the “one-day” cycle, renaming the symbols to start over again

$$r_{67,f,g} \equiv X_{f,g}[ ]_2^+ \longrightarrow [X_{f,g}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{68,f,g} \equiv \bar{X}_{f,g}[ ]_2^+ \xrightarrow{ps \cdot pg} [\bar{S}_{f,g,2} S_{f,g}^{20}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{69,f,g} \equiv \bar{X}_{f,g}[ ]_2^+ \xrightarrow{pg \cdot (1-ps)} [\bar{A}_{f,g,2} A_{f,g}^{20} \tilde{X}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{70,f,g} \equiv \bar{X}_{f,g}[ ]_2^+ \xrightarrow{1-ps} [X_{f,g}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{71,f,g,i} \equiv \bar{S}_{f,g,i}[ ]_2^+ \longrightarrow [\bar{S}_{f,g,i+1} S_{f,g}^{20}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 4 \end{cases}$$

$$r_{72,f,g,i} \equiv \bar{A}_{f,g,i}[ ]_2^+ \longrightarrow [\bar{A}_{f,g,i+1} A_{f,g}^{20} \tilde{X}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 4 \end{cases}$$

$$r_{73,f,g} \equiv \bar{A}_{f,g,5} [ ]_2^+ \longrightarrow [R_{f,g} \tilde{X}]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{74,f,g} \equiv \bar{S}_{f,g,5} [ ]_2^+ \xrightarrow{pr_g} [R_{f,g} \tilde{X}]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{75,f,g} \equiv \bar{S}_{f,g,5} [ ]_2^+ \xrightarrow{1-pr_g} [\#]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{76,f,g} \equiv R_{f,g} [ ]_2^+ \longrightarrow [R_{f,g} \tilde{X}]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

Clock reset with charge change

$$r_{77} \equiv [C_{18}]_2^+ \longrightarrow [C_1]_2$$

The constants associated with the rules have the following meaning:

- $q_{f,g,j}$ : Number of individuals in family  $f$ , in age group  $g$  inside community  $j$ .
- $l_{n,j}$ : Infected individuals in neighbourhood  $n$  of community  $j$ .
- $p_g$ : Probability for a person in age group  $g$  in contact with the virus to become infected.
- $pr_g$ : Probability for an infected person in age group  $g$  to recover.
- $ps$ : Probability for an infected person to be symptomatic.

The proposed model consists of seven modules of rules. The first module (infection of people) will only be executed when the model is initialized. The six remaining modules will be executed in a loop. Each cycle of the loop is interpreted as one day in the scenario.

In this chapter, a SIR model has been presented. In this model, the population is structured in age groups. Moreover, different contact spaces have been defined. The basic SIR model groups all individuals who are in a common location in a single age group. The features of PDP models permits the introduction of features for a more accurate reflection of reality in a straightforward manner.

## 4.6.2 Results

In the previous section, a SIR computational model based on Population Dynamics P systems has been presented. In order to evaluate the accuracy of the model with respect to the phenomenon under study, a validation process has been performed. The inherent randomness in complex systems like the one presented makes it infeasible the formal validation of models that attempt to reproduce their behaviour. It is therefore necessary an experimental validation by comparison of results generated by simulation tools with experimental data obtained directly from the real system. For this purpose, several software simulations have been performed, making use of

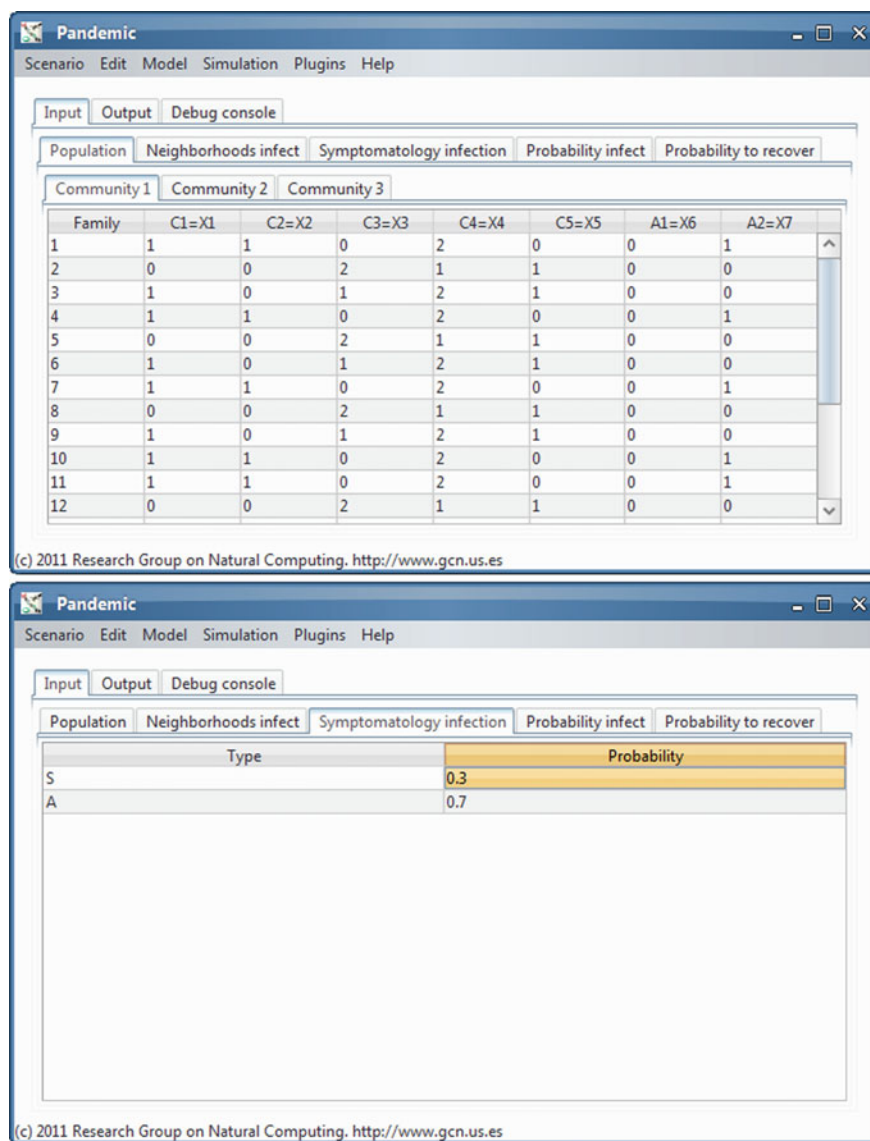


Fig. 4.5 MeCoSim Pandemic custom app—Input parameters tabs

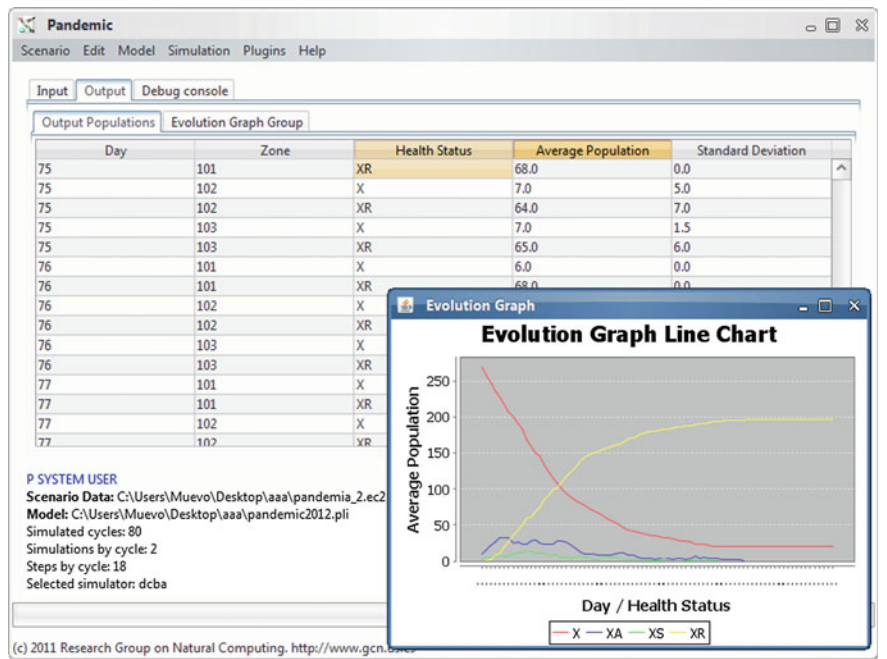
the tools presented in Sect. 4.5; that is, P-Lingua and the pLinguaCore library [21], providing a standard language to define P systems and a Java library to manage P-Lingua files and simulate P system computations, and MeCoSim [25], providing a visual environment to perform the simulations.



An application for SIR (called Pandemic) has been supplied with MeCoSim by customization. Thus, by simply defining a configuration file, a visual GUI has been provided, adapted to the parameters required for the presented model. The interested reader can find in [36] the MeCoSim application files which define the model and instructions to reproduce the experiments.

A number of virtual experiments has been performed by providing the general model for SIR in P-Lingua format, and then introducing the appropriate values for the data corresponding to different scenarios in the input tables of the MeCoSim window (see Fig. 4.5). The process for each given scenario is as follows: the input data are introduced, the corresponding parameters  $q_{f,g,j}$ ,  $l_{n,j}$ ,  $p_g$ ,  $pr_g$  and  $ps$  are generated, and then the computation is performed with Simulate! option, which calls DCBA-based simulation engine in pLinguaCore library. The simulation results have been obtained in the form of output tables and charts, as shown in Fig. 4.6.

A first scenario was simulated for the presented model. The detailed description of the scenario is what follows. The number of communities and families has been given as input parameters. These parameters have been obtained from [37]. The example depicted in Fig. 4.2 consists of 3 communities and 20 families for each community. At the initial stage, there are six infected people both in communities 1 and 3. In contrast, there are no infected people in community 2. The probability for a susceptible person



**Fig. 4.6** Pandemic—Output table and chart. Number of healthy, asymptomatic, symptomatic and recovered individuals by zone

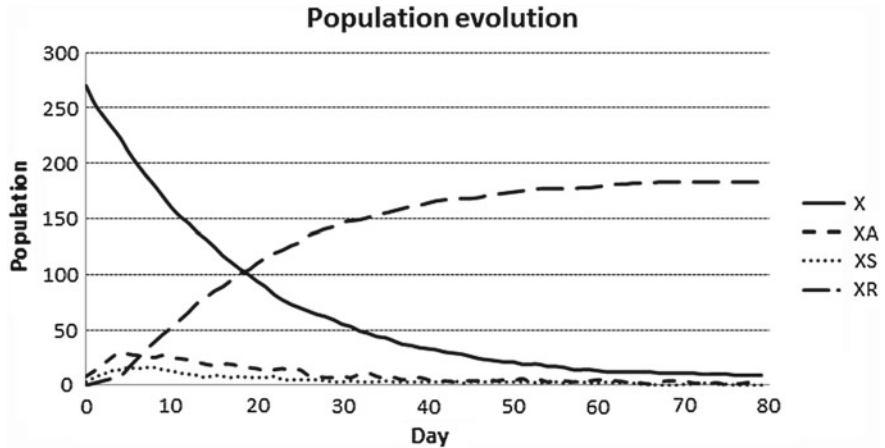


Fig. 4.7 Evolution of individuals—Original scenario

in contact with an infected one to become infected is 5 %. There is a 30 % probability for an infected person to manifest symptoms, whereas in the rest of the cases the symptoms are not manifested. Those who manifest symptoms are solely breeding grounds inside their family. On the other hand, those who do not manifest symptoms might infect people in other families and communities. There is a 5 % probability for a symptomatic person to recover. In contrast, asymptomatic people always recover. There exist no observable differences in the pandemic dynamics among the three communities due to the quick spreading of the disease. The designed simulator supports tuning of the parameters in the model. This feature provides a friendly way to study the behaviour of the disease under different scenarios. The presented model

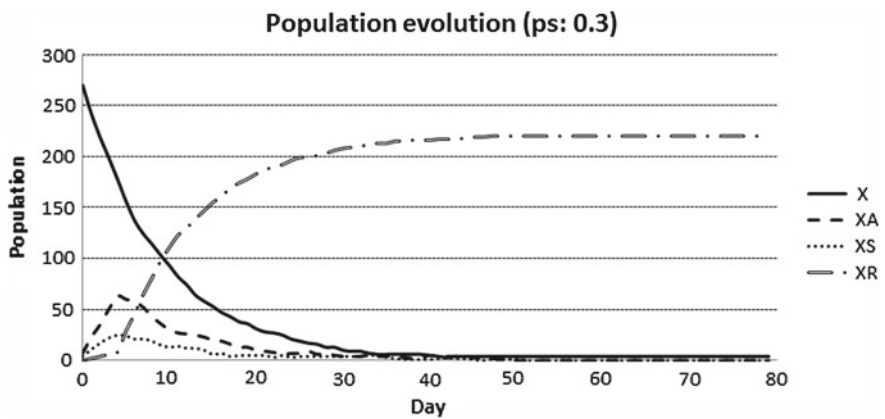


Fig. 4.8 Evolution of individuals—First alternative case

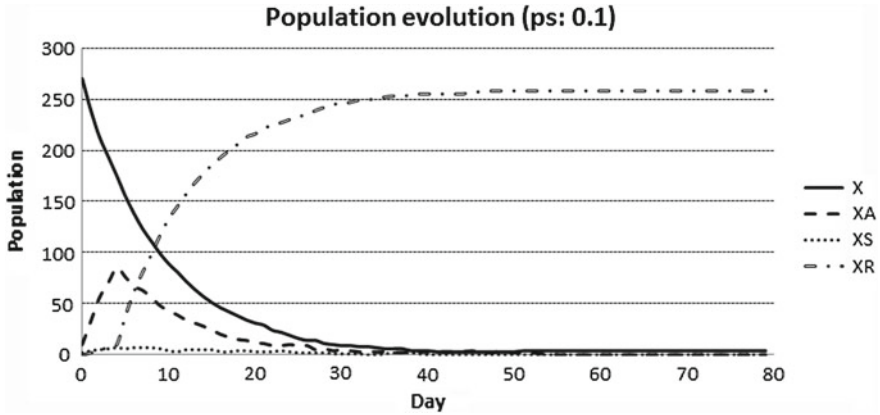


Fig. 4.9 Evolution of individuals—Second alternative case

does not take into account human interventions, such as vaccination campaigns, prophylactic actions, etc. These kind of measures are usually undertaken in the event of a pandemic outbreak. The results for this first scenario are shown in Fig. 4.7.

A number of other virtual experiments have been performed. Two of them are shown below. Both of them have similar input parameters; in particular, the probability for an individual to become infected is 10 %, and the probability of recovery is 30 % (the same for each age group). The only difference is the probability for an infected individual to be symptomatic. This probability is 30 % in the **first case** (see Fig. 4.8), whereas it is 10 % in the **second case** (see Fig. 4.9). As it can be seen, the second scenario presents a bigger number of recovered individuals, given the fact that asymptomatic individuals always recover.

Although the experiments carried out in this chapter refer to a virtual population, the results obtained by our simulations match the tendencies reported in the literature using classical SIR models [37–39].

## 4.7 Conclusions and Perspectives

Population Dynamics P systems (PDP systems) provide a new formal bio-inspired modelling framework. This is a novel and expressive approach that overcomes some limitations of classical mathematical models while keeping the most important features of the studied phenomena.

We illustrate the modelling/simulating workflow by means of a SIR computational model. First of all comes the design of the model, capturing complex social networks and interactions between individuals by means of the hierarchical structure of membranes (and the graph of connections between environments) along with their associated rules.

The model is then described in P-Lingua and given as input to the pLinguaCore library, which parses the model description and checks for errors. It is worth noting that such a description does not include data related to specific scenarios, only the initial structure and the rules. In order to input scenario-specific data related to a virtual experiment, MeCoSim generates a custom GUI. This GUI defines data fields specifically related to the problem at hand for the user to input the data.

In order to simulate the scenario, an algorithm capturing the semantics of PDP systems is required (e.g. MeCoSim calls pLinguaCore to perform this task relying on its built-in simulators). In this chapter we have explained the Direct distribution based on Consistent Blocks Algorithm (DCBA), which performs a proportional distribution of objects among rules in accordance to their associated probabilities. The algorithm grants a fair distribution in the case of competition among rules (i.e. rules having overlapping left-hand sides). Finally, simulation results are displayed by means of data tables and charts.

One of the drawbacks of sequential simulators is the time they spend to simulate considerably large instances. Nevertheless, the parallel structure of DCBA algorithm (and of PDP systems) appoints it suitable for its implementation on parallel hardware architectures, such as computer grids and graphic cards. In this sense, we have introduced an existing CUDA-based simulator which takes advantage of the computational parallel power of GPU computing in order to accelerate PDP systems simulations.

As regards to perspectives, the ultimate goal of these models is to serve as automatic assistants on management decision taking. That is, to give information about the effects of plausible measures by simulating presumed scenarios derived from undertaking these measures. In this sense, it is essential that future versions of the model consider and assess the effects of human measures. They will also need to analyse and foresee future trends on the studied populations. This goal calls for tight collaboration between experts and model designers, so as to define virtual experiments leading to feasible hypothesis to be verified by means of field work and live experiments.

When it comes to software development, the main work lines have to do with the improvement of the existing simulators, to reach higher performance and match even better the experimental results. As a short-term perspective, it is required to defined and implement efficient communication protocols to connect the mentioned parallel simulators from PMCGPU with the framework pLinguaCore. The addition of new functionalities to the interfaces is another important task which concerns software development. As a result, future versions of MeCoSim should permit a more exhaustive analysis on the results and augment the degree of automation of the design and simulation workflow from an end-user perspective.

**Acknowledgments** The authors acknowledge the support of “Proyecto de Excelencia con Investigador de Reconocida Valía” of the “Junta de Andalucía” under grant P08-TIC04200, and the support of the project TIN2012-37434 of the “Ministerio de Economía y Competitividad” of Spain, both co-financed by FEDER funds.

## References

1. P.J.E. Goss, J. Peccoud, Quantitative modelling of stochastic system in molecular biology by using stochastic Petri nets. *Proc. Nat. Acad. Sci. U.S.A.* **95**, 6750–6755 (1998)
2. A. Regev, E. Shapiro, The  $\pi$ -calculus as an abstraction for biomolecular systems, in *Modelling in Molecular Biology*, ed. by G. Ciobanu, G. Rozenberg (Springer, Berlin, 2004), pp. 219–266
3. A. Regev, E.M. Panina, W. Silvermann, L. Cardelli, E. Shapiro, BioAmbients: an abstraction for biological compartments. *Theoret. Comput. Sci.* **325**, 141–167 (2004)
4. L. Cardelli, Brane calculi: interactions of biological membranes. *Lect. Notes Bioinf.* **3082**, 257–278 (2005)
5. V. Danos, C. Laneve, Formal molecular biology. *Theoret. Comput. Sci.* **325**(1), 69–110 (2004)
6. D. Harel, Statecharts: a visual formalism for complex systems. *Sci. Comput. Program.* **8**(3), 231–274 (1987)
7. M. Holcombe, M. Gheorghe, N. Talbot, A hybrid machine model of rice blast fungus, *magnaphorte grisea*. *BioSystems* **68**(2–3), 223–228 (2003)
8. M.L. Shaffer, Determining minimum viable population sizes for the grizzly bear, in *Proceedings International Conference on Bear Research and Management* vol. 5, pp. 133–139 (1983)
9. M.E. Soulé (ed.), *Viable Populations for Conservation* (Cambridge University Press, Cambridge, 1987)
10. Gh. Păun, Computing with membranes, *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000) and Turku Center for Computer Science-TUCS, Report No 208
11. M. Cardona, M.A. Colomer, M.J. Pérez-Jiménez, D. Sanuy, A. Margalida, Modeling ecosystem using P systems: the bearded vulture, a case study. *Lect. Notes Comput. Sci.* **5391**, 137–156 (2009)
12. M.A. Colomer, A. Margalida, D. Sanuy, M.J. Pérez-Jiménez, A bio-inspired computing model as a new tool for modeling ecosystems: the avian scavengers as a case study. *Ecol. Model.* **222**(1), 33–47 (2011)
13. M. Cardona, M.A. Colomer, A. Margalida, A. Palau, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy, A computational modeling for real ecosystems based on P systems. *Nat. Comput.* **10**(1), 39–53 (2011)
14. M.A. Colomer, A. Montori, I. Gaspa, C. Fondevilla, A computational model to study the dynamics of Pyrenean Newt (*Calotriton asper*), in *Twelfth International Conference on Membrane Computing (CMC12)*, ed. by M. Gheorghe, Gh. Păun, S. Verlan, pp. 485–496 (2011)
15. Gh. Păun, *Membrane Computing: An Introduction* (Springer-Verlag, Berlin, 2002)
16. M.A. Colomer, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez, A uniform framework for modeling based on P systems, in *IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2010)*, vol. 1, ed. by K. Li, Z. Tang, R. Li, A.K. Nagar, R. Thamburaj, pp. 616–621 (2010)
17. S.E. Jørgensen, *Ecological Modelling: An introduction* (WIT press, Boston, 2009)
18. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M. García-Quismondo, L.F. Macías-Ramos, L. Valencia-Cabrera, A. Romero-Jiménez, C. Graciani, A. Riscos-Núñez, M.A. Colomer, M.J. Pérez-Jiménez, DCBA: simulating population dynamics P systems with proportional objects distribution. *Lect. Notes Comput. Sci.* **7762**, 257–276 (2013)
19. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez, F. Sancho-Caparrini, A simulation algorithm for multienvironment probabilistic P systems: a formal verification. *Int. J. Found. Comput. Sci.* **22**(1), 107–118 (2011)
20. D. Díaz-Pernil, C. Graciani, M.A. Gutiérrez-Naranjo, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Software for P systems, in *The Oxford Handbook of Membrane Computing*, Chapter 17, ed. by Gh. Păun, G. Rozenberg, A. Salomaa, (Oxford University Press, Oxford, 2009), pp. 437–454
21. I. Pérez-Hurtado, Desarrollo y aplicaciones de un entorno de programación para computación celular: P-Lingua. Ph.D. Thesis, University of Seville, 2010
22. The P-Lingua website. <http://www.p-lingua.org>
23. GNU Public License. <http://www.gnu.org/licenses/gpl.html>

24. M. Cardona, M.A. Colomer, A. Margalida, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy, A P system based model of an ecosystem of some scavenger birds. *Lect. Notes Comput. Sci.* **5957**, 182–195 (2010)
25. I. Pérez-Hurtado, L. Valencia-Cabrera, M.J. Pérez-Jiménez, M.A. Colomer, A. Riscos-Núñez. MeCoSim: a general purpose software tool for simulating biological phenomena by means of P systems, in *IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2010)*, vol. I, ed. by K. Li, Z. Tang, R. Li, A.K. Nagar, R. Thamburaj, pp. 637–643 (2010)
26. MeCoSim website. <http://www.p-lingua.org/mecosim>
27. M.A. Martínez-del-Amor, I. Karlin, R.E. Jensen, M.J. Pérez-Jiménez, A.C. Elster. Parallel simulation of probabilistic P systems on multicore platforms, in *Tenth Brainstorming Week on Membrane Computing (BWMC 2012)*, vol. II, ed. by M. García, L.F. Macías, Gh. Păun, L. Valencia, pp. 17–26 (2012)
28. M. Harris, *Mapping Computational Concepts to GPUs* (ACM SIGGRAPH 2005 Courses, NY, 2005)
29. F. Cabarle, H. Adorna, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez, Improving GPU simulations of Spiking Neural P systems. *Rom. J. Inform. Sci. Technol.* **15**(1), 5–20 (2012)
30. J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Simulation of P systems with active membranes on CUDA. *Briefings Bioinf.* **11**(3), 313–322 (2010)
31. J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Simulating a P system based efficient solution to SAT by using GPUs. *J. Logic Algebraic Program.* **79**(6), 317–325 (2010)
32. D. Kirk, W. Hwu, *Programming Massively Parallel Processors: A Hands On Approach* (Morgan Kauffman, Boston, 2010)
33. Ø. Krog, A.C. Elster, Fast GPU-based fluid simulations using SPH. *Lect. Notes Comput. Sci.* **7134**, 98–108 (2012)
34. M.A. Martínez-del-Amor, I. Pérez-Hurtado, A. Gastalver-Rubio, A.C. Elster, M.J. Pérez-Jiménez, population dynamics P systems on CUDA. *Lect. Notes Bioinf.* **7605**, 247–266 (2012)
35. The PMCGPU project website. <http://sourceforge.net/p/pmcgpu>
36. The Pandemic model in MeCoSim. [http://www.p-lingua.org/mecosim/doc/case\\_studies/pandemics.html](http://www.p-lingua.org/mecosim/doc/case_studies/pandemics.html)
37. H. Yasuda, K. Suzuki, Measures against transmission of pandemic H1N1 influenza in Japan in 2009: simulation model. *Euro Surveill* **14**, 44 (2009), pii=19385
38. H.W. Hethcote, Three basic epidemiological models. *Appl. Math. Ecol.* **18**, 119–144 (1989)
39. M. Qiao, A. Liu, U. Forys, Qualitative analysis of the SICR epidemic model with impulsive vaccinations. *Math. Methods Appl. Sci.* **36**(6), 695–706 (2013)

## Chapter 5

# Membrane Systems and Tools Combining Dynamical Structures with Reaction Kinetics for Applications in Chronobiology

Thomas Hinze, Jörn Behre, Christian Bodenstein, Gabi Escuela,  
Gerd Grünert, Petra Hofstedt, Peter Sauer, Sikander Hayat and Peter Dittrich

**Abstract** This chapter addresses three coordinated chronobiological studies demonstrating the convergence of experimental observations, fine-grained spatio-temporal modelling, and predictive simulation. Due to the discrete manner of molecular assembly in cell signalling and gene regulation, we define a framework of

---

T. Hinze (✉) · P. Hofstedt · P. Sauer  
Institute of Computer Science and Information and Media Technology, Brandenburg  
University of Technology, Postfach 10 13 44, 03013 Cottbus, Germany  
e-mail: thomas.hinze@tu-cottbus.de

P. Hofstedt  
e-mail: hofstedt@tu-cottbus.de

P. Sauer  
e-mail: peter.sauer@tu-cottbus.de

T. Hinze · G. Escuela · G. Grünert · P. Dittrich · C. Bodenstein  
School of Biology and Pharmacy, Department of Bioinformatics, Friedrich Schiller  
University of Jena, Ernst-Abbe-Platz 1–4, 07743 Jena, Germany  
e-mail: gabi.escuela@uni-jena.de

G. Grünert  
e-mail: gerd.gruenert@uni-jena.de

P. Dittrich  
e-mail: peter.dittrich@uni-jena.de

C. Bodenstein  
e-mail: christian.bodenstein@uni-jena.de

J. Behre  
Theoretical Systems Biology Group, Institute of Food Research, Norwich Research Park,  
Colney, Norwich NR4 7UA, UK  
e-mail: joern.behre@ifr.ac.uk

S. Hayat  
Department of Biochemistry and Biophysics, University of Stockholm,  
Svante Arrhenius väg 16C, SE-106 91 Stockholm, Sweden  
e-mail: shayat@cbr.su.se

membrane systems equipped with discretised forms of reaction kinetics in concert with variable intramolecular structures. Our first study is dedicated to circadian clocks inducing daily biological rhythms. As an inspiring example, the KaiABC core oscillator reaches its functionality by cyclically varying protein structures. Within our second study, we present a meta-model of an entire circadian clockwork able to adapt its oscillation to an external stimulus in terms of a frequency control system acting in a phase-locked loop. Substrate concentration courses resulting from gene expression reflect its oscillatory behaviour utilised in a periodical trigger for subsequent processes. In this context, our third study cytometrically quantifies the dynamical behaviour of a bistable toggle switch resulting from mutual gene regulation.

## 5.1 Introduction

In recent years, chronobiology emerged as a promising research area with various exploitable applications in medicine, agriculture, and biotechnology [36]. It is focused on endogenous biological and chemical rhythms acting as trigger signals and for control of manifold subsequent processes. A prominent example is given by circadian clocks which comprise chemical oscillatory systems with a free-running periodicity close to 24 h. Their capability of entrainment to dedicated external stimuli like daily variations of sunlight and darkness makes circadian clocks a precise and robust nanoscaled frequency control system affecting numerous metabolic and gene regulatory activities. In the human body for instance, the intensity of alertness and tiredness has been influenced in this way mainly dependent on the temporal course of specific hormone and protein concentrations. Insufficiencies within such oscillatory systems can cause serious diseases. A detailed modelling of underlying processes at a molecular scale in conjunction with capturing its complex interplay at a systems level is an essential prerequisite to cope with the challenge of achieving a thorough, holistic understanding of biological rhythms.

The framework of membrane systems provides an ideal modelling approach in chronobiology since it can combine a rule-based description of dynamical spatial structures with time-dependent kinetic issues. In this context, each single molecule within a reaction system including its submolecular structures, attached ligands, and complex formations can be considered and traced over time with its structural modifications according to configurable reaction kinetics. We introduce the resulting class of string-based P systems which includes definition of cell signalling P modules, spatially delimited functional reaction units for (bio)chemical information processing. This introduction comes along with a presentation of the corresponding simulation software package SRSim which incorporates spatial rules and a strong visualisation engine. Moreover, three examples illustrate diverse application scenarios of our framework in chronobiology together with descriptive simulation studies.

Starting with the KaiABC core oscillator [64] found in the procaryotic cyanobacterium *Synechococcus elongatus*, one of the oldest life forms on earth, we explore a cell signalling network representing a post-translational prototype for generation



of circadian rhythms. It turns out that this oscillator includes an inherent frequency divider which allows release of trigger signals every 12h instead of 24h. In the second scenario, we extend a generalised core oscillator by additional modules in order to identify functional components useful for adaptation to an external oscillatory signal. We show that phase-locked loops provide an efficient way of chemical frequency control just by combining a controllable core oscillator with a chemical signal multiplier and a reaction cascade acting as a low-pass filter. The output of a circadian clock, the temporal course of specific molecular species concentrations, typically serves as a trigger for subsequent processes. This functionality often implies a chemical counterpart of logic switches and gates responsible for signal merge and evaluation. Our third application scenario is dedicated to an in-vivo implementation of a binarily operating NAND gate and a bistable toggle switch (flip-flop). Both units consist of gene regulatory networks present in the lac operon that exists in many life forms. Here, transcription factors as signalling molecules control the functionality of information processing.

## 5.2 The KaiABC Core Oscillator: A Circadian Clock Component with Dynamical Molecular Structures

### 5.2.1 Biological Background

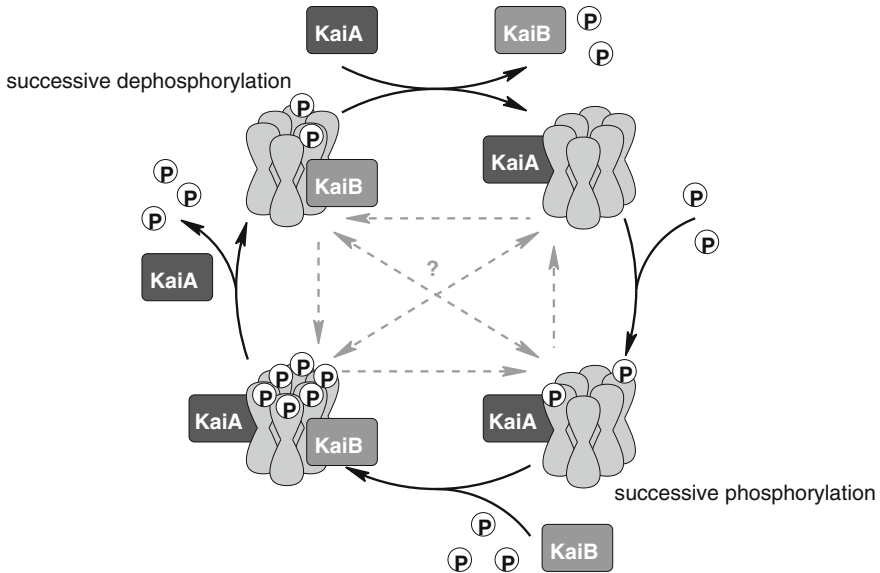
Circadian rhythms enclose an interesting biological phenomenon that can be seen as a widespread property of life [58]. The coordination of biological activities into daily cycles provides an important advantage for the fitness of diverse organisms [5, 60]. Based on self-sustained biochemical oscillations, circadian clocks are characterised by a natural period close to but slightly different from 24h that persists under constant conditions (like constant darkness or constant light). Their ability for compensation of temperature in the physiological range enables them to maintain the period in case of environmental changes. Furthermore, circadian clocks can be entrained [48]. This property allows a gradual reset of the underlying oscillatory system for adjustment by exposure to external stimuli like light/dark or temperature cycles. A variety of metabolic, cell signalling, and gene regulatory processes is synchronised or controlled by circadian clocks. Chemically, they utilise an individual cyclic reaction scheme including one or more feedback loops. Most of the circadian clocks comprise gene transcription and translation feedback loops [2].

Surprisingly, the procaryotic cyanobacterium *Synechococcus elongatus* was discovered to carry a post-translational circadian clock even functioning in vitro [64]. Three key clock proteins KaiA, KaiB, and KaiC could be identified together with their atomic structure [47]. KaiC as the focal protein rhythmically oscillates between hypophosphorylated and hyperphosphorylated forms [50]. The spatial structure of KaiC represents a homohexamer shaped as a “double doughnut” with 6 phosphorylation twin sites at the interfaces between monomeric subunits. Presence of the supple-

mentary protein KaiA specifically enhances KaiC phosphorylation while complex formation of KaiBC activates KaiC dephosphorylation [46]. The KaiABC circadian oscillator appears as a reaction cycle consisting of four consecutive phases [21], see Fig. 5.1: KaiAC complex formation releasing KaiB, successive KaiAC phosphorylation, KaiABC complex formation, and successive KaiABC dephosphorylation in conjunction with KaiA dissociation. Each of these phases takes approximately 6 h. There is some evidence for further interactions between the aforementioned protein complexes and intermediate products in terms of negative feedback loops stabilising the oscillation. However, the detailed mechanism is still unclear and gives room for hypotheses translated into a couple of candidate models [5]. In this section, we reflect and extend our results initially discussed in Ref. [32].

### 5.2.2 Membrane Systems $\Pi_{\text{CSM}}$ for Cell Signalling Modules

Biological signalling networks have been identified to exhibit a universal capability to process information [31, 42]. They can be viewed as complex computational devices of the cell, triggering and directing responses to external stimuli. It turns out that successive formation or decomposition of protein complexes in conjunction with



**Fig. 5.1** Reaction cycle of the KaiABC oscillator characterised by four phases and incomplete information about interphase feedback loops, arranged from descriptions of the oscillatory mechanism given in Refs. [21, 46]. A corresponding minimal model of the four-phase cycle has been proposed in Ref. [5]

domain-specific protein binding (e.g. phosphorylation by kinases) plays a central role in biological signal transduction based on submolecular assembly [3]. In this context, resulting biomolecules act as information carriers of astonishing storage capacity and structural plasticity. For example, the tumor suppressor protein p53 is equipped with 27 phosphorylation sites [4]. It could theoretically carry up to  $2^{27} = 134,217,728$  different activation states. Having in mind that each of these states is able to form an individual constituent of a reaction network incorporating all distinguishable states of up to several hundred interacting proteins, the potential dimension of those protein signalling networks is obvious.

While the steady-state behaviour might be sufficient to characterise a metabolic network (cf. [28]), the function of a protein signalling network depends heavily on its temporal evolution. Oscillators based on phosphorylation/dephosphorylation cycles represent significant examples [46, 50, 64]. Thus, the aspect of *dynamical behaviour* should be reflected in the choice of the preferred modelling approach. For that purpose, ordinary differential equations (ODEs) derived from appropriate kinetics are commonly applied. Since this method usually assumes each individual protein activation state to act as a separate species, it easily leads to an exponential growth of the number of distinct ODEs (addressed amongst others in Ref. [12]). An opportunity to temporarily unify several activation states by one dedicated species could be a keystone to overcome this insufficiency.

Inspired by this initial idea, we propose a P systems framework able to specify proteins together with relevant properties by string-objects. In contrast to species names in ODEs, phenotypic information about a protein is represented by a character string. Each individual protein property is allowed to be marked as present, absent, or arbitrary. In the latter case, placeholders known from regular expressions denote unassigned protein properties. Consequently, reaction rules may also contain placeholders processed by a matching relation for association of available particles to reactants given within rules. Furthermore, our P systems framework combines the ability to manage specific string-objects with discretised reaction kinetics. Incomplete information about protein activation states can be handled by setting placeholders if required. While they enable a unification of several activation states when specifying a protein on the one hand, placeholders, on the other hand, contribute to trace the variety of potential effects by embedding wild-cards into reaction rules. Thus, a bottom-up strategy for the modelling of signalling networks by successive knowledge integration can benefit from the proposed framework. Along with intermediate results coming from simulation of a partially wild-carded system, synergies between wetlab experimental setup and model refinement considering structural dynamics might emerge. Inclusion of reaction kinetics into the formalism of P systems was explained in Ref. [43], exemplified by metabolic networks, supplemented by signalling and gene regulatory networks [29]. A previous formulation of periodic and quasi-periodic processes based on symbol objects without inner structure is given in Ref. [9]. The BioNetGen framework [11] allows handling of string pattern to constitute species. However, its expressive capability of reaction kinetics excludes stoichiometry.

The mathematical specification of membrane systems is mainly based on *multisets* whose objects commonly symbolise single molecules able to interact within a dedicated reaction system [17, 51, 52]. Rewriting rules formally operating on multisets emulate the effect of chemical reactions on the underlying multiset of molecular objects. Let  $A$  be an arbitrary set,  $\mathbb{N}$  the set of natural numbers including zero, and  $\mathbb{R}_+$  the set of non-negative real numbers.  $\mathcal{P}(A)$  denotes the power set of  $A$ . A *fractional multiset* over  $A$  is a mapping  $F : A \rightarrow \mathbb{R}_+ \cup \{+\infty\}$ .  $\lfloor F(a) \rfloor$ , also denoted as  $[a]_F$ , specifies the multiplicity of  $a \in A$  in  $F$ . Particularly, ordinary multisets  $F : A \rightarrow \mathbb{N} \cup \{+\infty\}$  exclusively utilise whole-number cardinalities. Multisets in general can be written as an elementwise enumeration of the form  $\{(a_1, F(a_1)), (a_2, F(a_2)), \dots\}$  since  $\forall (a, b_1), (a, b_2) \in F : b_1 = b_2$ . The support  $\text{supp}(F) \subseteq A$  of  $F$  is defined by  $\text{supp}(F) = \{a \in A \mid F(a) > 0\}$ . A multiset  $F$  over  $A$  is said to be empty iff  $\forall a \in A : F(a) = 0$ . The cardinality  $|F|$  of  $F$  over  $A$  is  $|F| = \sum_{a \in A} F(a)$ . Let  $F_1$  and  $F_2$  be multisets over  $A$ .  $F_1$  is a subset of  $F_2$ , denoted as  $F_1 \subseteq F_2$ , iff  $\forall a \in A : (F_1(a) \leq F_2(a))$ . Multisets  $F_1$  and  $F_2$  are equal iff  $F_1 \subseteq F_2 \wedge F_2 \subseteq F_1$ . The intersection  $F_1 \cap F_2 = \{(a, F(a)) \mid a \in A \wedge F(a) = \min(F_1(a), F_2(a))\}$ , the multiset sum  $F_1 \uplus F_2 = \{(a, F(a)) \mid a \in A \wedge F(a) = F_1(a) + F_2(a)\}$ , and the multiset difference  $F_1 \ominus F_2 = \{(a, F(a)) \mid a \in A \wedge F(a) = \max(F_1(a) - F_2(a), 0)\}$  form multiset operations. Multiplication of a multiset  $F = \{(a, F(a)) \mid a \in A\}$  with a scalar  $c$ , denoted  $c \cdot F$ , is defined by  $\{(a, c \cdot F(a)) \mid a \in A\}$ . The term  $\langle A \rangle = \{F : A \rightarrow \mathbb{R}_+ \cup \{+\infty\}\}$  describes the set of all fractional multisets over  $A$ .

### 5.2.2.1 Definition of System Components

A membrane system for a cell signalling module (CSM) is a construct

$$\Pi_{\text{CSM}} = (V, V', R_1, \dots, R_r, f_1, \dots, f_r, A, C, \Delta\tau) \quad (5.1)$$

where  $V$  and  $V'$  are two alphabets (not necessarily disjoint); without loss of generality  $\#, \neg, \star \notin V \cup V'$ . The regular set

$$S = V^+ \cdot (\{\#\} \cdot ((V')^+ \cup \{\neg\} \cdot (V')^+ \cup \{\star\}))^*$$

describes the syntax for string objects. The leftmost substring from  $V^+$  holds the protein identifier, followed by a finite number of protein property substrings from  $(V')^+$  which are separated by  $\#$ . For example, consider the string object  $C: D\#p\#\#\neg q$  identifying protein (complex)  $C:D$  with specified property  $p$ , a second arbitrary property  $(\star)$ , and without property  $q$ . Each protein property substring expresses a specific additional information about the protein, for instance whether it is activated by carrying a ligand at a certain binding site. Two kinds of meta symbols are allowed. The symbol  $\neg$  excludes the subsequent property but permits all other properties at this substring position. The placeholder  $\star$  stands for an arbitrary (also unknown or unspecified) protein property substring. This way, uncertainty about the properties of proteins can be explicitly expressed. String objects can be dynamically processed by reaction rules:

$R_i \in \langle S \rangle \times \langle S \rangle$	is a reaction rule composed of two finite multisets
$f_i : \langle S \rangle \longrightarrow \mathbb{R}_+$	is a function corresponding to kinetics of reaction $R_i$
$A \in \langle S \rangle$	is a multiset of axioms representing the initial molecular configuration
$C \in \mathbb{R}_+ \setminus \{0\}$	spatial capacity of the module (vessel or compartment)
$\Delta\tau \in \mathbb{R}_+ \setminus \{0\}$	time discretisation interval

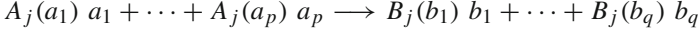
We explain the system evolution of  $\Pi_{\text{CSM}}$  within four consecutive subsections. Based on the specification of the system configuration, we define an iteration scheme that updates this configuration from time  $t$  to time  $t + 1$ . The update includes processing of reactions given by the rules  $R_i$  ( $i = 1, \dots, r$ ). For this purpose, an appropriate matching between wild-carded strings representing reactants and those stated in the current configuration is required. Then, a reaction is executed by removing the multiset of matching reactants from the current configuration followed by adding the corresponding products. In order to consider kinetic issues, each reaction can be multiply processed. Therefore, the number of turns is provided by the function  $f_i$ . Please note that  $\Pi_{\text{CSM}}$  evolves deterministically in contrast to stochastic approaches like Gillespie [19] or Master Equation [20].

### 5.2.2.2 Dynamical System Behaviour

A membrane system of the form  $\Pi_{\text{CSM}}$  evolves by successive progression of its *configuration*  $L_t \in \langle S \rangle$  at discrete points in time  $t \in \mathbb{N}$  for what we assume a global clock. Two consecutive dates  $t$  and  $t + 1$  specify a time span  $\Delta\tau$  (discretisation interval). A system step at time  $t$  consists of two modification stages per reaction  $1, \dots, r$ . First, the multiset of reactants is determined and removed from  $L_t$ . Afterwards, the corresponding multiset of products is added. To cope with conflicts that can occur if the available amount of reactants cannot satisfy all matching reactions, we prioritise the reaction rules by their index:  $R_1 > R_2 > \dots > R_r$ . Thus, we keep determinism of the system evolution and have mass conservation.

$$\begin{aligned}
 L_0 &= L_{0,0} = A \\
 L_{t,1} &= \begin{cases} L_{t,0} \ominus \text{Reactants}_{t,1} \uplus \text{Products}_{t,1} & \text{if } \text{Reactants}_{t,1} \subseteq L_{t,0} \\ L_{t,0} & \text{otherwise} \end{cases} \\
 L_{t,2} &= \begin{cases} L_{t,1} \ominus \text{Reactants}_{t,2} \uplus \text{Products}_{t,2} & \text{if } \text{Reactants}_{t,2} \subseteq L_{t,1} \\ L_{t,1} & \text{otherwise} \end{cases} \\
 &\vdots \\
 L_{t+1} = L_{t,r} &= \begin{cases} L_{t,r-1} \ominus \text{Reactants}_{t,r} \uplus \text{Products}_{t,r} & \text{if } \text{Reactants}_{t,r} \subseteq L_{t,r-1} \\ L_{t,r-1} & \text{otherwise} \end{cases}
 \end{aligned}$$

Let  $R_j = (A_j, B_j) \in \langle S \rangle \times \langle S \rangle$  be a reaction rule with  $\text{supp}(A_j) = \{a_1, \dots, a_p\}$  and  $\text{supp}(B_j) = \{b_1, \dots, b_q\}$ . In terms of a chemical denotation, it can be written as



where  $A_j(a_1), \dots, A_j(a_p)$  represent stoichiometric factors of reactants  $a_1, \dots, a_p$ , and  $B_j(b_1), \dots, B_j(b_q)$  stoichiometric factors of products  $b_1, \dots, b_q$ , respectively. All reactant strings that match to the pattern  $a_k$  are provided by a dedicated relation  $Match(a_k)$  (see Sect. 5.2.2.3 for definition). A combination of reactant strings from  $L_t$  matching the left hand side of  $R_j$  forms a multiset of string objects used to apply the reaction once. Since the kinetic law, described by the corresponding scalar function  $f_j$ , returns the number of applications of reaction rule  $R_j$  within one step, the multiset of string objects extracted from  $L_t$  to act as reactants for  $R_j$  can be written as  $Reactants_{t,j}$ :

$$Reactants_{t,j} = \biguplus_{e_1 \in Match(a_1)} \dots \biguplus_{e_p \in Match(a_p)} f_j(\{(e_1, \infty), \dots, (e_p, \infty)\} \cap L_{t,j-1}) \cdot \{(e_1, A_j(a_1)), \dots, (e_p, A_j(a_p))\}$$

Accordingly, the multiset of products resulting from reaction rule  $R_j$  is determined by the multiset  $Products_j(t)$ :

$$Products_{t,j} = \biguplus_{e_1 \in Match(a_1)} \dots \biguplus_{e_p \in Match(a_p)} f_j(\{(e_1, \infty), \dots, (e_p, \infty)\} \cap L_{t,j-1}) \cdot \{(b_1, B_j(b_1)), \dots, (b_q, B_j(b_q))\}$$

### 5.2.2.3 Matching

Let the regular set  $S$  be a syntax description for string objects. In the symmetric relation  $Match$ , two string objects match iff there is at least one common representation without wild-cards. This loose strategy requires a minimum degree of similarity between objects with incomplete information. Uncertainty is interpreted as arbitrary replacements within the search space given by  $S$ .

$$Match \subseteq S \times S$$

$$Match = \bigcup_{m \in \mathbb{N}} \{(p \# p_1 \# p_2 \dots \# p_m, s \# s_1 \# s_2 \dots \# s_m) \mid (p = s) \wedge \forall j \in \{1, \dots, m\} : [(p_j = s_j) \vee (p_j = \star) \vee (s_j = \star) \vee ((p_j = \neg q) \wedge (s_j \neq q)) \vee ((s_j = \neg q) \wedge (p_j \neq q))]\}$$

Matching of a single string object  $w \in S$  to the entire set  $S$  is defined by

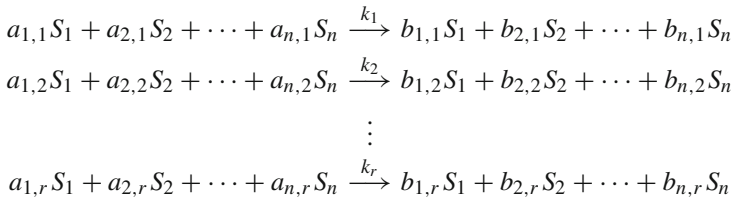
$$Match(w) = \{s \in S \mid (w, s) \in Match\}$$

Consequently, we define the matching of a language  $L \subseteq S$  by the function  $Match : \mathcal{P}(S) \longrightarrow \mathcal{P}(S)$  with

$$Match(L) = \bigcup_{w \in L} Match(w).$$

#### 5.2.2.4 Mass-action Kinetics

For description of the temporal behaviour of chemical reaction networks we consider substrate concentrations over time presuming homogeneity in reaction space. General mass-action kinetics [14] formulates reaction system's dynamics subject to production and consumption rates  $v_p$  and  $v_c$  of each substrate  $S$  in order to continuously change its concentration by  $\frac{d[S]}{dt} = v_p - v_c$ . A reaction system with a total number of  $n$  substrates and  $r$  reactions



employs stoichiometric factors  $a_{i,j} \in \mathbb{N}$  (reactants),  $b_{i,j} \in \mathbb{N}$  (products) and kinetic constants  $k_j \in \mathbb{R}_{>0}$  assigned to each reaction quantifying its velocity ( $\mathbb{N}$ : natural numbers,  $\mathbb{R}_{>0}$ : positive real numbers). The corresponding ODEs read [15]:

$$[\dot{S}_i] = \frac{d[S_i]}{dt} = \sum_{h=1}^r \left( k_h \cdot (b_{i,h} - a_{i,h}) \cdot \prod_{l=1}^n [S_l]^{a_{l,h}} \right) \quad \text{with } i = 1, \dots, n$$

In order to obtain a concrete trajectory, all initial concentrations  $[S_i](0) \in \mathbb{R}_{\geq 0}$ ,  $i = 1, \dots, n$  are allowed to be set according to the relevance for the reaction system.

#### 5.2.2.5 Time-discrete Reaction Kinetics

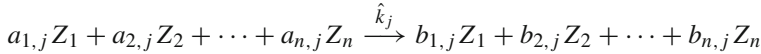
Within the P systems framework  $\Pi_{\text{CSM}}$ , we formulate reaction kinetics by specification of scalar functions  $f_j$  attached to corresponding reactions  $R_j$  ( $j = 1, \dots, r$ ). Each scalar function converts the current configuration  $L_t$ , a multiset of string objects, into the corresponding amount for update of reactant and product abundance with respect to rewriting rule  $R_j$ :

$$f_j(L_t) = k_j \prod_{\forall \alpha \in Match(A_j) \cap Match(L_t) : (R_j = (A_j, B_j))} \hat{f}(L_t(\alpha))^{|Match(A_j) \cap \{(\alpha, \infty)\}|} \quad (5.2)$$

whereas the auxiliary term  $\alpha$  passes through all string objects present in  $L_t$  which also form reactants in  $R_j$ . The multiplicity  $L_t(\alpha)$  of occurrences of  $\alpha$  acts as argument for a kinetic law  $\hat{f}(L_t(\alpha))$ . Examples adopted from mass-action, Michaelis–Menten, and Hill kinetics are shown in Fig. 5.2.

### 5.2.2.6 Relations to ODE-based Reaction Kinetics

For a reaction system with a total number of  $n$  species ( $i = 1, \dots, n$ ) and  $r$  reactions ( $j = 1, \dots, r$ )



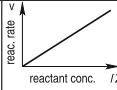
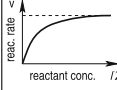
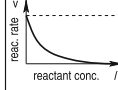
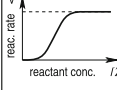
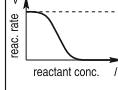
the corresponding ODEs

$$\frac{d[Z_i]}{dt} = \sum_{j=1}^r \left( \hat{k}_j \cdot (b_{i,j} - a_{i,j}) \cdot \prod_{l=1}^n \hat{f}_j([Z_l])^{a_{l,j}} \right) \quad \text{with } i = 1, \dots, n. \quad (5.3)$$

describe the temporal systems behaviour by consideration of stoichiometric coefficients  $a_{i,j} \in \mathbb{N}$  (reactants) and  $b_{i,j} \in \mathbb{N}$  (products) as well as a kinetic law  $\hat{f}_j([Z_i]) : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  that maps a species concentration  $[Z_i]$  into an effective reaction rate [14]. All initial concentrations  $[Z_i](0) \in \mathbb{R}_+$ ,  $i = 1, \dots, n$  are allowed to be set according to the needs of the reaction system.

A species concentration  $[Z_i] := \frac{z_i}{C}$  is defined as fraction of its molecular amount  $z_i = \text{card}(\{(Z_i, z_i)\}) = |Z_i|$  with respect to the spatial system capacity  $C \in \mathbb{R}_+$ .

A correspondence between the reaction rate  $k_j$  (employed in  $\Pi_{\text{CSM}}$  by function  $f_j$  attached to reaction  $R_j$ ) and the kinetic constant  $\hat{k}_j$  utilised in ODE (5.3) can be obtained by the Euler method of integrating differential equations. Discretisation of (5.3) with respect to time results in:

Kinetics	Activation	Inhibition
Mass-Action (no saturation)	 $\hat{f}([Z]) = [Z]$	—
Michaelis-Menten (saturation)	 $\hat{f}([Z]) = \frac{[Z]}{\Theta + [Z]}$	 $\hat{f}([Z]) = \left(1 - \frac{[Z]}{\Theta + [Z]}\right)$
Higher-Order Hill (saturation)	 $\hat{f}([Z]) = \frac{[Z]^n}{\Theta + [Z]^n}$	 $\hat{f}([Z]) = \left(1 - \frac{[Z]^n}{\Theta + [Z]^n}\right)$

**Fig. 5.2** Overview of several widely used kinetic laws  $\hat{f}([Z])$  dependent on reactant concentration  $[Z]$ . Parameters: threshold  $\Theta \in \mathbb{R}_+ \setminus \{0\}$ , Hill coefficient  $n \in \mathbb{N} \setminus \{0\}$



$$\frac{z_{i,t+1} - z_{i,t}}{\Delta\tau} = \sum_{j=1}^r \left( \hat{k}_j \cdot (b_{i,j} - a_{i,j}) \cdot \prod_{l=1}^n \hat{f}_j([Z_l])^{a_{l,j}} \right)$$

$$z_{i,t+1} - z_{i,t} = C \cdot \Delta\tau \cdot \sum_{j=1}^r \left( \hat{k}_j \cdot (b_{i,j} - a_{i,j}) \cdot \prod_{l=1}^n \hat{f}_j([Z_l])^{a_{l,j}} \right)$$

By setting  $k_j = \hat{k}_j \cdot C \cdot \Delta\tau$ , we obtain:

$$z_{i,t+1} - z_{i,t} = k_1(b_{i,1} - a_{i,1}) \prod_{l=1}^n \hat{f}_1([Z_l])^{a_{l,1}} + \cdots + k_r(b_{i,r} - a_{i,r}) \prod_{l=1}^n \hat{f}_r([Z_l])^{a_{l,r}}$$

Replacing  $k_j \cdot \hat{f}_j([Z_l])^{a_{l,j}}$  by the scalar function  $f_j(L_t)$  from Eq. (5.2) leads to:

$$z_{i,t+1} - z_{i,t} = (b_{i,1} - a_{i,1}) \cdot f_1(L_t) + \cdots + (b_{i,r} - a_{i,r}) \cdot f_r(L_t)$$

Since the stoichiometric coefficients  $a_{i,j}$  and  $b_{i,j}$  of each reaction  $R_j = (A_j, B_j)$  in  $\Pi_{\text{CSM}}$  are expressed by multisets  $A_j$  (reactants) and  $B_j$  (products), we write:

$$z_{i,t+1} - z_{i,t} = (B_1(b_i) - A_1(a_i)) \cdot f_1(L_t) + \cdots + (B_r(b_i) - A_r(a_i)) \cdot f_r(L_t)$$

From that, we achieve the update scheme for species  $Z_i$  present in  $L_t$  with  $z_{i,t}$  copies at time  $t$  by processing reaction  $R_j$ :

$$z_{i,t+1} = z_{i,t} - A_j(Z_i) \cdot f_j(L_t) + B_j(Z_i) \cdot f_j(L_t)$$

By extension from a single species to the entire configuration along with inclusion of matching, we finally obtain

$$L_{t+1,j} = L_{t,j} \ominus \text{Reactants}_{t,j} \uplus \text{Products}_{t,j}$$

in accordance to the iteration scheme for  $\Pi_{\text{CSM}}$  evolution.

### 5.2.3 Applying $\Pi_{\text{CSM}}$ to a KaiABC Core Oscillator Model

We identify a cell signalling module for the cyclic reaction scheme sketched in Fig. 5.1. Key proteins KaiA, KaiB, and KaiC resulting from expression of corresponding genes are assumed to be present in the module ab initio. Considering the core oscillator, 17 reaction rules along with loose matching correspond to the four-phase reaction cycle. Successive KaiC phosphorylation in the presence of KaiA is expressed by rules  $R_1$ – $R_6$  followed by successive dephosphorylation in the presence of KaiB within rules  $R_7$ – $R_{12}$ . Finally,  $R_{13}$  and  $R_{14}$  formulate inhibiting KaiA/KaiB exchange acting as negative feedback loops, and  $R_{15}$ – $R_{17}$  reflect protein degradation. A kinetic function  $f$  is attached to each reaction rule that follows from time-discrete

Michaelis–Menten kinetic laws in concert with linear mass-action kinetics for protein degradation.

$$\Pi_{KaiABC} = (V, V', R_1, \dots, R_{17}, f_1, \dots, f_{17}, A, C, \Delta\tau)$$

$$V = \{C\} \cup \dots \text{identifier of the focal protein KaiC}$$

$$\{A, B\} \dots \text{identifiers of proteins KaiA and KaiB}$$

$$V' = \{A, B\} \cup \dots \text{KaiA, KaiB within a complex associated to KaiC}$$

$$\{\varepsilon, P, P^2, P^3, P^4, P^5, P^6\} \dots \text{number of attached phosphates}$$

$$R_1 = C\# \neg A\# B\# \varepsilon + A \longrightarrow C\# A\# \neg B\# P^1 + B$$

$$R_i = C\# A\# * \# P^{i-1} + A \longrightarrow C\# A\# * \# P^i + A, \quad i = 2, \dots, 5$$

$$R_6 = C\# A\# \neg B\# P^5 + B \longrightarrow C\# \neg A\# B\# P^6 + A$$

$$R_{i+7} = C\# * \# B\# P^{i+1} + B \longrightarrow C\# * \# B\# P^i + B, \quad i = 0, \dots, 5$$

$$R_{13} = C\# \neg A\# B\# * + A \longrightarrow C\# A\# \neg B\# * + B$$

$$R_{14} = C\# A\# \neg B\# * + B \longrightarrow C\# \neg A\# B\# * + A$$

$$R_{15} = A \longrightarrow \emptyset$$

$$R_{16} = B \longrightarrow \emptyset$$

$$R_{17} = C\# * * * * \longrightarrow \emptyset$$

$$f_1(L_t) = k_1 \cdot \frac{L_t(C\# \neg A\# B\# \varepsilon)}{\Theta_{1,1} + L_t(C\# \neg A\# B\# \varepsilon)} \cdot \frac{L_t(A)}{\Theta_{1,2} + L_t(A)}$$

$$f_i(L_t) = k_i \cdot \frac{L_t(C\# A\# * \# P^{i-1})}{\Theta_{i,1} + L_t(C\# A\# * \# P^i)} \cdot \frac{L_t(A)}{\Theta_{i,2} + L_t(A)}, \quad i = 2, \dots, 5$$

$$f_6(L_t) = k_6 \cdot \frac{L_t(C\# A\# \neg B\# P^5)}{\Theta_{6,1} + L_t(C\# A\# \neg B\# P^5)} \cdot \frac{L_t(B)}{\Theta_{6,2} + L_t(B)}$$

$$f_{i+7}(L_t) = k_{i+7} \cdot \frac{L_t(C\# * \# B\# P^{i+1})}{\Theta_{i+7,1} + L_t(C\# * \# B\# P^i)} \cdot \frac{L_t(B)}{\Theta_{i+7,2} + L_t(B)}, \quad i = 0, \dots, 5$$

$$f_{13}(L_t) = k_{13} \cdot \left(1 - \frac{L_t(C\# \neg A\# B\# *)}{\Theta_{13,1} + L_t(C\# \neg A\# B\# *)}\right) \cdot \left(1 - \frac{L_t(A)}{\Theta_{13,2} + L_t(A)}\right)$$

$$f_{14}(L_t) = k_{14} \cdot \left(1 - \frac{L_t(C\# A\# \neg B\# *)}{\Theta_{14,1} + L_t(C\# A\# \neg B\# *)}\right) \cdot \left(1 - \frac{L_t(B)}{\Theta_{14,2} + L_t(B)}\right)$$

$$f_{15}(L_t) = k_{15} \cdot L_t(A)$$

$$f_{16}(L_t) = k_{16} \cdot L_t(B)$$

$$f_{17}(L_t) = k_{17} \cdot L_t(C\# * * * *)$$

$$A \in \{C\# * * * *\}$$

### 5.2.4 Simulation Case Study

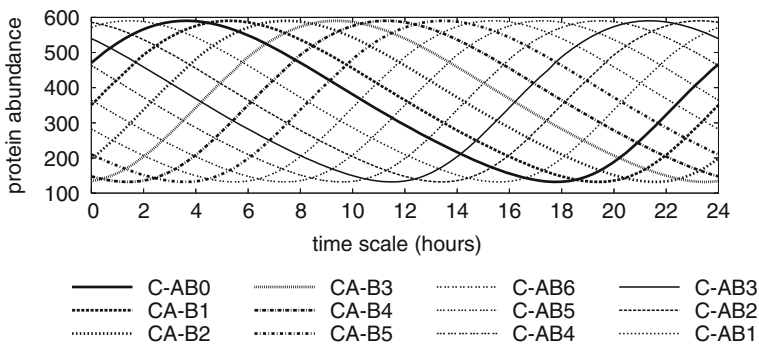
Using the KaiABC circadian oscillator we conducted a simulation case study to demonstrate the practicability of the modelling approach addressed before. The reaction scheme formulated by the P system  $\Pi_{KaiABC}$  exhibits a high degree of symmetry among its constituents. The main reaction cycle is composed of 12 consecutive feed-forward reactions flanked by widespread negative feedback loops. They affect each intermediate product within the reaction cycle following the intention of an inhibiting KaiA/KaiB exchange independent of the phosphorylation state.

For simulation of the dynamical behaviour of  $\Pi_{KaiABC}$ , we empirically parameterise and initialise the system in a symmetric way to obtain phase-shifted protein abundance courses which stably oscillate with a period of approximately 24 h. To avoid a transient oscillation phase, the initial amounts of protein constituents were set directly at the discrete limit cycle. This constraint is reflected in the following multiset of axioms:

$$A = \{(C\# \neg A\#B\#\varepsilon, 470), (C\#A\# \neg B\#P^1, 351), (C\#A\# \neg B\#P^2, 198), \\ (C\#A\# \neg B\#P^3, 135), (C\#A\# \neg B\#P^4, 148), (C\#A\# \neg B\#P^5, 210), \\ (C\# \neg A\#B\#P^6, 282), (C\# \neg A\#B\#P^5, 364), (C\# \neg A\#B\#P^4, 463), \\ (C\# \neg A\#B\#P^3, 541), (C\# \neg A\#B\#P^2, 586), (C\# \neg A\#B\#P^1, 571), \\ (A, 2,520), (B, 2,520)\}$$

Each KaiC protein within the pattern  $C\# * * * \#$  keeps an average amount of 360 copies (arbitrarily chosen).

Figure 5.3 shows the corresponding individual protein abundance courses resulting from following parameter setting for the discrete iteration scheme:  $\Theta_{i,1} = 79.2$ ,

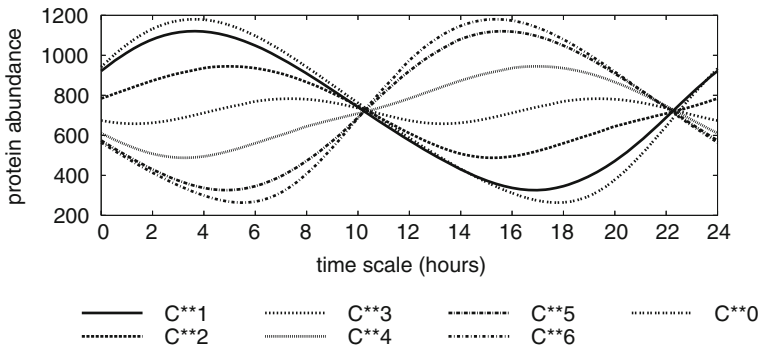


**Fig. 5.3** Temporal courses of 12 specific KaiABC subproducts representing the process status of the reaction cycle. Kinetic parameters and initial amounts adjusted in a way to obtain a period of  $\approx 24$  h and symmetry among individual oscillations

$\Theta_{i,2} = 554.4, \hat{k}_i = 360.0$  for  $i \in \{1, \dots, 12\}$ ;  $\Theta_{i,1} = 64.8, \Theta_{i,2} = 453.6, \hat{k}_i = 412.8$  for  $i \in \{13, 14\}$ , and  $\hat{k}_{15} = \hat{k}_{16} = 508.1, \hat{k}_{17} = 254.6$ ;  $C = 1.2, \Delta\tau = 0.05$ .

Based on the individual protein abundance courses depicted in Figs. 5.3, 5.4 illustrates the effect of subsuming KaiABC subproducts according to their number of attached phosphates ranging from 0 to 6. Association of KaiA and KaiB is neglected here resulting in consideration of regular expressions  $C\#*\#*P^i$  for  $i = 0, \dots, 6$ . The simulation shows that medium phosphorylation levels possess smaller amplitudes than minor or major phosphorylation levels. Due to symmetry reasons, KaiABC subproducts carrying three phosphates double the frequency of oscillation. Hence, the reaction system is able to act as a frequency scaler. This feature could be useful to control downstream processes at a subcircadian granularity. The protein abundance courses obtained by our simulation approximately reproduce experimental results [49].

The simulation case study demonstrates the practicability of membrane systems for cell signalling modules. Our framework  $\Pi_{\text{CSM}}$  intends to combine advantages of processing regular expressions that represent molecular entities with the corresponding dynamical behaviour of an entire reaction network resulting from superpositioning of individual molecular abundance courses. To this end, we have integrated string objects into a deterministic framework able to emulate time-discrete forms of reaction kinetics in concert with dedicated matching strategies in order to identify reactants from the current system configuration. From an algebraic point of view, oscillations that occur in structural or configural dynamics of P systems can be detected using a backtracking mechanism along with the temporal system evolution: By monitoring the overall configurations over time, a derivation tree is obtained. Stable oscillations appear as recurring, but non-adjacent overall configurations along a path through the derivation tree.



**Fig. 5.4** Temporal courses of KaiABC subproducts subsumed by their level of phosphorylation ranging from 0 to 6. Kinetic parameters and initial amounts adjusted in a way to obtain a period of  $\approx 24$  h and symmetry among individual oscillations

### 5.3 Circadian Clocks as Generalised Frequency Control Systems

There are numerous biochemical core oscillators found in living organisms' circadian clocks. From a systems biology point of view, this is no surprise since a simple cyclic reaction scheme containing at least one feedback loop can suffice to exhibit a sustained oscillatory behaviour. Furthermore, each core oscillator must be able to vary its frequency according to a dedicated tuning signal produced by an upstream reaction system in order to adapt its signal course to an external stimulus. Probably a plethora of evolutionary origins led to a variety of circadian core oscillators. From today's perspective, the majority of them reveals sinusoidal or almost sinusoidal signal courses as exemplified by the KaiABC core oscillator in the previous Section. These core oscillators have mainly in common a multi-phased reaction cycle whose enzymatically catalysed stages induce a mixture of activating and inhibiting effects. Interestingly, the absolute number of individual stages forming the entire cycle just slightly influences the oscillatory waveform which is typically close to a sinusoidal behaviour. This enables a gradual and smooth alteration such that the transfer between minimal and maximal signal levels consumes a notable amount of time. In most cases, the oscillation passes a stable limit cycle which acts as an attractor. This feature makes the oscillator quite robust against perturbations affecting the signal course. In this context, the so-called *Goodwin-type* core oscillator emerged to represent a generalised meta-model for consideration of circadian clocks based on sinusoidal or almost sinusoidal waveforms.

Circadian clock systems appear to be special forms of frequency control systems. Following this line, it should be possible to identify appropriate interacting modules representing elements of a dedicated control-loop model. Indeed, coupling of a controllable core oscillator with a low-pass filter and a multiplier succeeds to reproduce the desired entrainment behaviour of a circadian clock. This Section aims at consistent formulation of a generalised circadian clock model derived from an underlying chemical frequency control system introduced in Ref. [33].

#### 5.3.1 A Controllable Goodwin-Type Core Oscillator

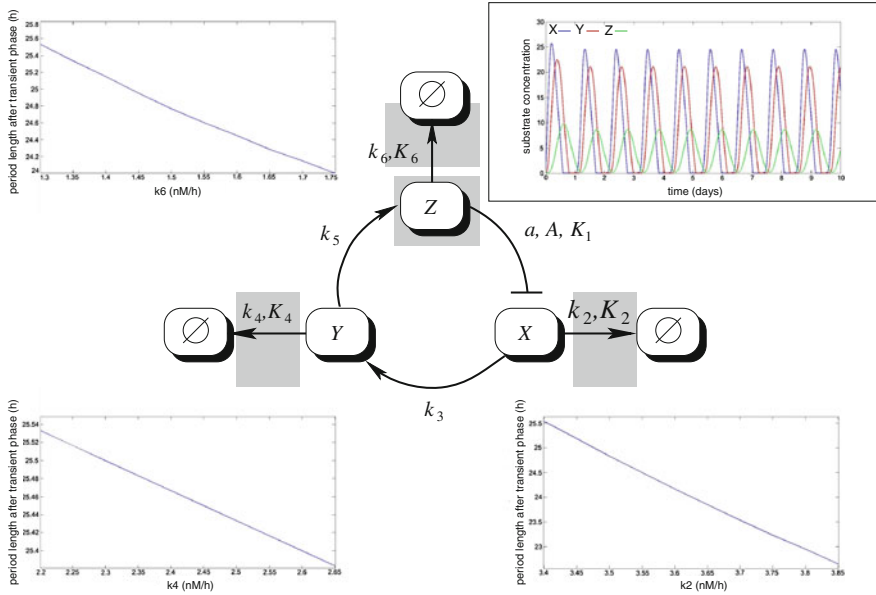
Let us first consider a controllable Goodwin-type core oscillator on its own. A Goodwin-type oscillator comprises an abstract model of a cyclic gene regulatory network, which is able to exhibit a sustained oscillatory behaviour in its substrate concentrations [22]. Goodwin-type oscillators have in common three distinct substrates typically called  $X$ ,  $Y$ , and  $Z$ . All components degrade in the presence of specific proteases acting as catalysts. It turns out that the velocity of degradation is the most effective way to control the oscillation frequency [61].

Here, we apply a version of the Goodwin-type core oscillator model, which utilises Michaelis–Menten kinetics [14] instead of mass-action kinetics for degradation due to its saturational nature. Additionally, we take into account the transportation of a substrate into the nucleus of the cell. This transportation can be commonly seen

as a prerequisite for downstream process control. Interestingly, the resulting model reaches sustained oscillations with low Hill coefficients  $h$  in Hill terms of the form  $\frac{\alpha}{\beta + \gamma[S]^h}$  (values  $\geq 1$  suffice here, we choose 2). Overall, we obtain the membrane system:

$$\begin{aligned}
 \Pi_{Goodwin} &= (V, V', R_1, \dots, R_6, f_1, \dots, f_6, A, C, \Delta\tau) \\
 V &= \{X, Y, Z\} \\
 V' &= \emptyset \\
 R_1 &= Z \longrightarrow X \\
 R_2 &= X \longrightarrow \emptyset \\
 R_3 &= X \longrightarrow Y \\
 R_4 &= Y \longrightarrow \emptyset \\
 R_5 &= Y \longrightarrow Z \\
 R_6 &= Z \longrightarrow \emptyset \\
 f_1(L_t) &= \frac{a}{A + K_1 \cdot L_t(Z)^2} \\
 f_2(L_t) &= \frac{k_2 \cdot L_t(X)}{K_2 + L_t(X)} \\
 f_3(L_t) &= k_3 \cdot L_t(X) \\
 f_4(L_t) &= \frac{k_4 \cdot L_t(Y)}{K_4 + L_t(Y)} \\
 f_5(L_t) &= k_5 \cdot L_t(Y) \\
 f_6(L_t) &= \frac{k_6 \cdot L_t(Z)}{K_6 + L_t(Z)}
 \end{aligned}$$

Figure 5.5 reveals the behaviour of our Goodwin-type core oscillator with respect to its incorporation into a frequency-control loop. The upper-right diagram gives a notion of the waveforms generated by the oscillator, which become visible after a short transient phase of approx. 1.5 days. It turns out that  $X$  and  $Y$  demonstrate a widely sinusoidal course emerging from a ground plateau. Moreover, the course of  $Z$  resembles a nearly perfect sinusoidal shape. This advantageous feature makes the oscillatory signal of  $Z$  easier than the others to compare with an oscillatory external stimulus. Hence, the temporal concentration course of  $Z$  exhibits the best fit to act as output signal. Supplementary diagrams within Fig. 5.5 show the almost linear dependency of the oscillator's period length on the velocity parameters  $k_2$ ,  $k_4$ , and  $k_6$  of the corresponding degradation reactions. Therefore, *regulated degradation* by proteases affecting one or more of these velocity parameters can control the oscillatory frequency, which is the reciprocal of the period length.



**Fig. 5.5** Topology and behaviour of a Goodwin-type core oscillator. Its oscillation frequency can be most effectively controlled by the velocity parameters  $k_2$ ,  $k_4$ , and  $k_6$  of the degradation reactions of substrates  $X$ ,  $Y$ , and  $Z$ . The upper-right diagram illustrates the oscillatory waveforms of  $X$ ,  $Y$ , and  $Z$  substrate concentrations; parameter setting:  $a = 6$ ,  $A = 0.6$ ,  $K_1 = 1$ ,  $K_2 = 0.2$ ,  $k_2 = 3.4$ ,  $k_3 = 0.3$ ,  $K_4 = 0.2$ ,  $k_4 = 2.2$ ,  $k_5 = 0.1$ ,  $K_6 = 1.44$ ,  $k_6 = 1.3$ ,  $L_0(X) = 1$ ,  $L_0(Y) = L_0(Z) = 0$ ,  $C = 20$ ,  $\Delta\tau = 0.05$ . The initial concentrations of  $Y$  and  $Z$  were intentionally set to 0 to demonstrate that the oscillator reaches its limit cycle after a short transient phase, which emphasises the stability of the oscillatory system. After the transient phase, we obtain limit cycle oscillations whose free-running period can vary between approx. 22 and 26 h

### 5.3.2 Chemical Frequency Control by Phase-Locked Loops

Originally, control loops had been introduced in engineering to achieve a desired dynamical behaviour of a system like adjusting temperature or local time of a clock [8]. We can distinguish between two different meanings of “adjustment”. On the one hand, it comprises the elimination of external perturbations. In this application scenario, a predefined reference specifies the desired dynamical behaviour while the influence of external stimuli has to be compensated. Here, variations of external stimuli are usually interpreted to be perturbations, for instance if temperature should be kept constant. On the other hand, a control loop can be constructed in order to adapt its dynamical behaviour to the course of an external stimulus. In this case, the deviance between the control loop’s output and the external stimulus has to be minimised. Imagine a radio controlled clock as an example. Throughout this contribution, we will consistently focus on the latter of both scenarios.

Numerous technical attempts succeeded in construction of an abundance of control loops operating mechanically, hydraulically, pneumatically, electrically,

electronically, or even chemically [8]. Their simplest scheme has a general closed feedback loop in common composed of four basic elements for real-valued signal processing: plant, sensor, controller, and actuator.

**Plant:** The plant (sometimes also called system) is constituted by one or more physical quantities whose temporal behaviour has been controlled. Its temporal input is given by a tuning signal  $v(t)$  which passes through the system leading to its output  $x(t) = P(v(t))$ . The transfer function  $P$  might include signal weakening, delay, or perturbation.

**Sensor:** The sensor transforms  $x(t)$  into the measured output  $y(t) = F(x(t))$  where  $F$  acts as transfer function. In some cases, the sensor is dispensable if the plant output can be processed directly holding  $y(x) = x(t)$ .

**Controller:** The controller compares  $y(t)$  to the external stimulus (reference signal)  $w(t)$  and calculates the error signal  $e(t) = D(w(t), y(t))$ . Subsequently, it provides the control signal  $u(t) = C(e(t))$ . The underlying transfer function  $C$  might include integration or differentiation with respect to  $t$ .

**Actuator:** The actuator affects the plant by transforming  $u(t)$  into the tuning signal  $v(t) = A(u(t))$ , which feeds back to the plant.

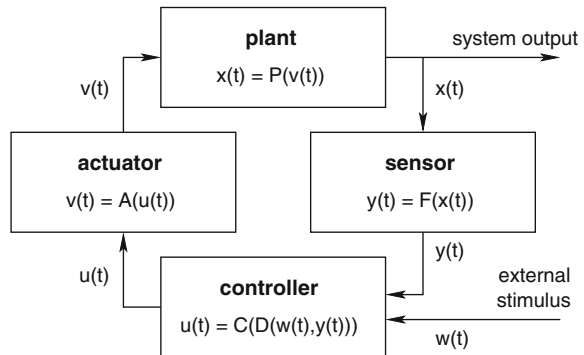
Signal processing is commonly represented by block diagrams containing characteristic curves or transfer functions like  $P$ ,  $F$ ,  $D$ ,  $C$ , and  $A$  that map input or memorised signals into output signals (cf. Fig. 5.6).

Later, control loops came into the scope of life sciences as part of a cybernetic approach to understand biological systems, now preferably called *control systems* [26]. They benefit from a strict modularisation that allows a clear decomposition of a complex system into interconnected signal-processing units [41, 65].

From a systems point of view, circadian clocks constitute biochemical regulatory circuits whose functionality resembles technical counterparts utilising phase-locked loops (PLLs) [10]. Corresponding circuits comprise three modules:

1. a *core oscillator (plant)* whose frequency has been controlled to adapt to an external stimulus. The intensity and periodicity of environmental light converted into a specific protein abundance by a photo cascade represents a typical external stimulus.

**Fig. 5.6** General scheme of a control loop





2. a *signal comparator* (**controller**), for instance a phase detector, responsible for determining the deviation between the signal produced by the core oscillator on the one hand and the external stimulus on the other. The signal comparator carries out an arithmetic task. Interestingly, a single complex formation (dimerisation) conducting a multiplication in a mathematical manner succeeds here whereas also more sophisticated mechanisms could be involved.
3. a biochemical *low-pass filter* (**actuator**) completes the control system by providing a global feedback loop.

We construct a PLL-based chemical model of a frequency control system by composition of a Goodwin-type core oscillator (plant) with an arithmetic signal comparator (controller) and a low-pass filter (actuator). For the resulting entire system topology, see Fig. 5.10.

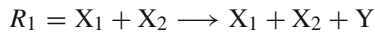
### 5.3.2.1 A Chemical Signal Comparator

The signal comparator is responsible for estimation of the *phase shift* between the oscillatory signal  $L_t(Z)$  released by the core oscillator and the external stimulus  $L_t(E)$  as reference signal. Interestingly, a simple arithmetic signal multiplication is sufficient if both oscillatory signals exhibit a sinusoidal or almost sinusoidal course, let us assume  $\sin(\omega_1 t)$  and  $\cos(\omega_2 t + \psi)$ . Multiplication leads to  $\sin(\omega_1 t) \cdot \cos(\omega_2 t + \psi) = \frac{1}{2} (\sin((\omega_1 - \omega_2)t - \psi) + \sin((\omega_1 + \omega_2)t + \psi))$  due to elementary trigonometric laws. While the term  $\sin((\omega_1 + \omega_2)t + \psi)$  exclusively comprises the frequency sum  $\omega_1 + \omega_2$ , it becomes eliminated by passing the low-pass filter and hence it can be neglected. The remaining term  $\sin((\omega_1 - \omega_2)t - \psi)$  indicates the estimated phase shift to be minimised. To do so, the external stimulus' phase becomes locked while the core oscillator has been forced to “catch it up” by temporarily enhancing its frequency until the phase shift is compensated. Our signal comparator utilises a simple chemical signal multiplier of the form:

$$\Pi_{\text{multiplier}} = (V, V', R_1, R_2, f_1, f_2, A, C, \Delta\tau)$$

$$V = \{X_1, X_2, Y\}$$

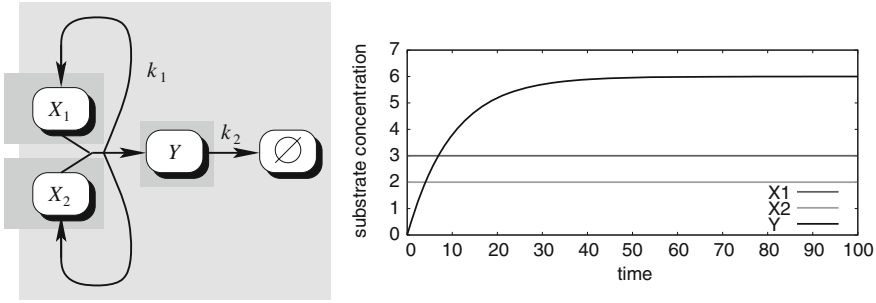
$$V' = \emptyset$$



$$f_1(L_t) = k_1 \cdot L_t(X_1) \cdot L_t(X_2)$$

$$f_2(L_t) = k_2 \cdot L_t(Y)$$

Let  $k_1 = k_2 > 0$ . A dimerisation suffices to emulate an arithmetic multiplication using mass-action kinetics. We assume the regeneration of consumed input substrates by an autocatalytic loop. Auxiliary side reactions can be added in order to assure mass conservation if needed. Figure 5.7 illustrates the module topology along with



**Fig. 5.7** “ $3 \cdot 2 = 6$ ”; parameter setting:  $k_1 = k_2 = 0.1$ ,  $L_0(X_1) = 3$ ,  $L_0(X_2) = 2$ ,  $L_0(Y) = 0$ ,  $C = 20$ ,  $\Delta\tau = 0.05$

an example. The system reaches its steady state comprising

$$\lim_{t \rightarrow \infty} L_t(Y) = L_0(X_1) \cdot L_0(X_2).$$

### 5.3.2.2 Low-pass Filter

Acting as a moving average element, a simple variant of a chemical low-pass filter can be found by a linear reaction cascade. The sequence of consecutively running reactions causes a successive delay along with smoothing of the passing signals. Each cascade stage buffers and accumulates the arriving molecules for a varying time span before transduction to the next stage. This time-limited accumulation is mainly responsible for the filtering effect on oscillatory input signals. While low frequency signals pass the filter, higher frequency oscillations become more and more diminished and hence eliminated. In addition, oscillatory waveforms undergo a conversion into a sinusoidal shape since higher-order harmonics get lost. The behaviour of a low-pass filter can be specified by a so-called Bode plot, which depicts the intensity of signal weakening subject to different frequencies. Interestingly, biochemical low- and band-pass filters had already been identified in signalling cascades of intracellular processes [44, 62]. Assuming a low-pass filter composed of  $n$  stages, an according P system reads:

$$\Pi_{lpf} = (V, V', R_1, R_2, \dots, R_n, R_{n+1}, f_1, f_2, \dots, f_n, f_{n+1}, A, C, \Delta\tau)$$

$$V = \{X, X_1, \dots, X_{n-1}, Y\}$$

$$V' = \emptyset$$

$$R_1 = X \longrightarrow X_1$$

$$R_2 = X_1 \longrightarrow X_2$$

$$\vdots$$

$$\begin{aligned}
R_{n-1} &= X_{n-2} \longrightarrow X_{n-1} \\
R_n &= X_{n-1} \longrightarrow Y \\
R_{n+1} &= Y \longrightarrow \emptyset \\
f_1(L_t) &= k_1 \cdot L_t(X) \\
f_2(L_t) &= k_2 \cdot L_t(X_1) \\
&\vdots \\
f_{n-1}(L_t) &= k_{n-1} \cdot L_t(X_{n-2}) \\
f_n(L_t) &= k_n \cdot L_t(X_{n-1}) \\
f_{n+1}(L_t) &= k_{n+1} \cdot L_t(Y)
\end{aligned}$$

The transfer function for non-oscillatory input signals is described by a smoothing delay. For oscillatory input signals apart from an analytical evaluation, the transfer function becomes replaced by a characteristic curve, which arises from simulation studies or measure. In some cases, preferably for sinusoidal signals and those entirely captured by a finite Fourier series, the temporal behaviour can be analytically mapped into a frequency domain using Laplace transform. A Bode plot describes the characteristic curve of a low-pass filter by pointing out two essential parameters, the *cutoff frequency* and the *slope*, see Fig. 5.9. The cutoff frequency marks the transition from the passband to the stopband whereas the signal amplitude becomes more and more weakened. Usually, the relative intensity of weakening is given logarithmically in magnitudes  $\text{dB} = 10 \cdot \log \left( \frac{\text{amplitude of output signal}}{\text{amplitude of input signal}} \right)$ . The slope characterises the increase of damping within the stopband. Once estimated, both parameters—cutoff frequency and slope—are sufficient from a systemic point of view to determine the behaviour of a low-pass filter instead of managing numerous kinetic constants and initial concentrations.

Figure 5.8 exemplarily depicts the effect of a five-stage low-pass filter to a poly-frequential input concentration course  $L_t(X)$  while Fig. 5.9 shows the corresponding Bode plot. The number of stages  $n$  defines the degree of asymptotic smoothing into a sinusoidal output concentration course after transient phase.

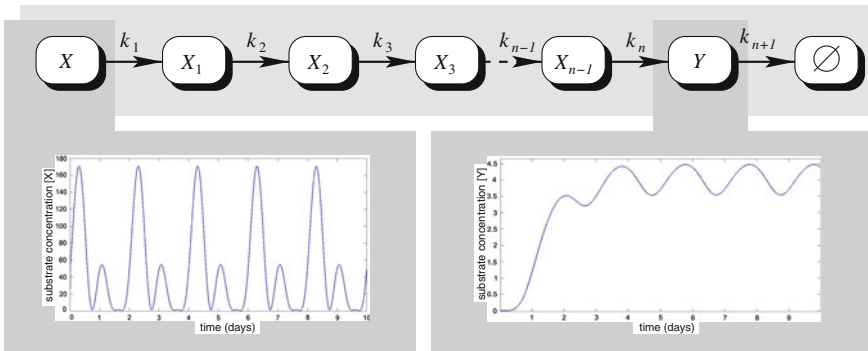
The chosen parameter setting of the low-pass filter within a PLL-based circadian clock model entails a cutoff frequency of approx.  $1.9 \cdot 10^{-5} \text{s}^{-1}$ , which corresponds to a period length of approx. 0.61 days in accordance with the need for a circadian clock system.

### 5.3.3 Exploring Circadian Clock's Entrainment Behaviour by Simulation Studies

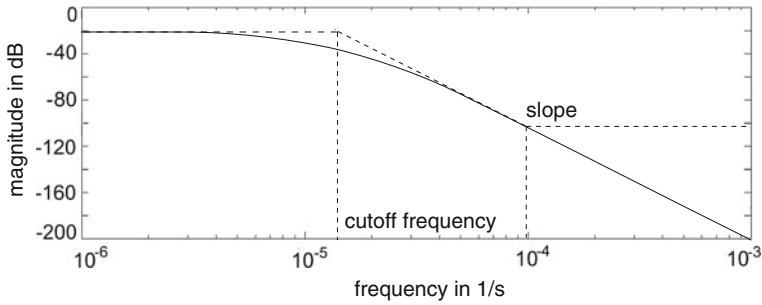
Having now the entire system as shown in Fig. 5.10 at hand, we conducted a couple of simulation studies to explore whether the overall behaviour of our control-system model coincides with expectations for circadian clocks.

In a first scenario, the external stimulus persists at a constant value  $L_t(E)$  according to permanent darkness or permanent light. We expect a so-called *free-running* sustained oscillation of  $L_t(Z)$  whose period might be influenced by the constant external stimulus. Figure 5.11a confirms this dependency by an almost linear relation. The period length becomes shortened by ascending values of the external stimulus while permanent darkness ( $L_t(E) = 0$ ) indicates the maximum period length (approx. 25 h30' using the above-mentioned parameter setting).

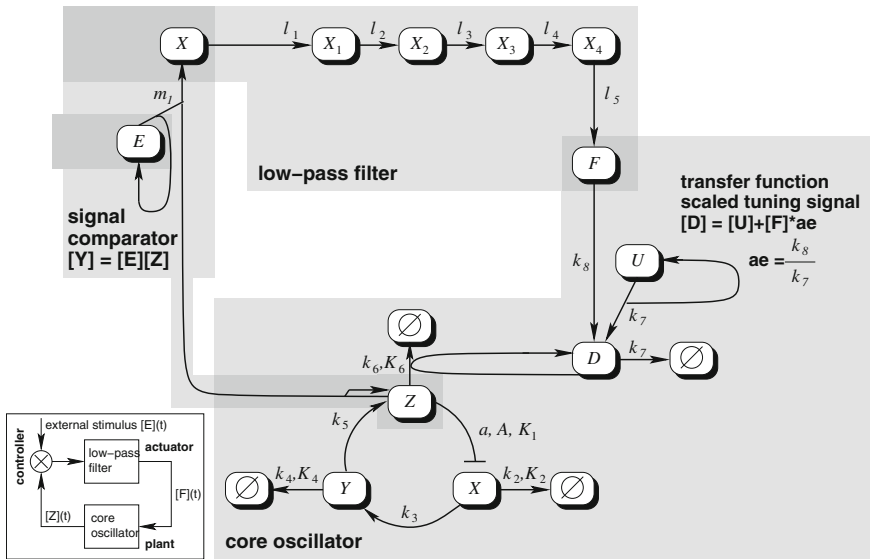
A second simulation study is inspired by gaining insight into the time to entrainment if  $L_t(E)$  and  $L_t(Z)$  start to oscillate with no phase shift but comprising distinct initial frequencies instead. We choose a sinusoidal course of the external stimulus  $L_t(E)$ . The core oscillator is configured to exhibit a natural period of 24 h12' while the period of the external stimulus  $L_t(E)$  might differ. Figure 5.11b plots the duration needed until the core oscillator's output  $L_t(Z)$  adapted to the period of  $L_t(E)$  (time to entrainment). We define that entrainment is reached if the period length of  $L_t(Z)$  converges to those of  $L_t(E)$  up to 1'. Entrainment which demands shortening of  $Z$ 's period is more efficient than enhancing. This is plausible in the context of how PLL-based control operates: In case of a phase shift between  $L_t(E)$  and  $L_t(Z)$ , the resulting error signal  $L_t(F)$  leads to a temporary acceleration of  $Z$  degradation, which in turn increases the oscillation frequency causing a shortened period. In contrast, entrainment to slower clock signals incorporates a form of inherent graduality: First,  $L_t(Z)$  temporarily increases its frequency until it converges to  $L_t(E)$  for a moment before it drifts again what causes a new adjustment of  $L_t(Z)$ . The process of successive adjustment repeats consistently in a time-consuming manner.



**Fig. 5.8** Simulation of a five-stage chemical low-pass filter affecting a poly-frequential input signal given by substrate concentration course  $L_t(X)$ . This input course was obtained by multiplication of two arbitrarily chosen sinusoidal signals  $f(t)$  and  $g(t)$  with periods of 1 and 2 days:  $f(t) = 7 + 6.9 \sin\left(\frac{2\pi}{1} \cdot t\right)$ ,  $g(t) = 7 + 6.9 \sin\left(\frac{2\pi}{2} \cdot t\right)$  resulting in  $L_t(X) = f(t) \cdot g(t) = 49 + 48.3 \sin\left(\frac{2\pi}{1} \cdot t\right) + 48.3 \sin\left(\frac{2\pi}{2} \cdot t\right) + 23.805 \cos\left(\frac{2\pi}{1} \cdot t\right) + 23.805 \cos\left(\frac{2\pi}{2} \cdot t\right)$ . The output course  $L_t(Y)$  reveals the filtering effect by providing a monofrequential signal of almost sinusoidal shape with period length of 2 days after transient phase. The higher frequency signal with period length of 1 day became eliminated; parameter setting:  $k_1 = k_2 = k_3 = k_4 = 0.036$ ,  $k_5 = 3,600$ ,  $k_6 = 180$ ,  $L_0(X_i) = 0$  for  $i = 1, \dots, 4$ ,  $L_0(Y) = 0$ ,  $C = 20$ ,  $\Delta\tau = 0.05$

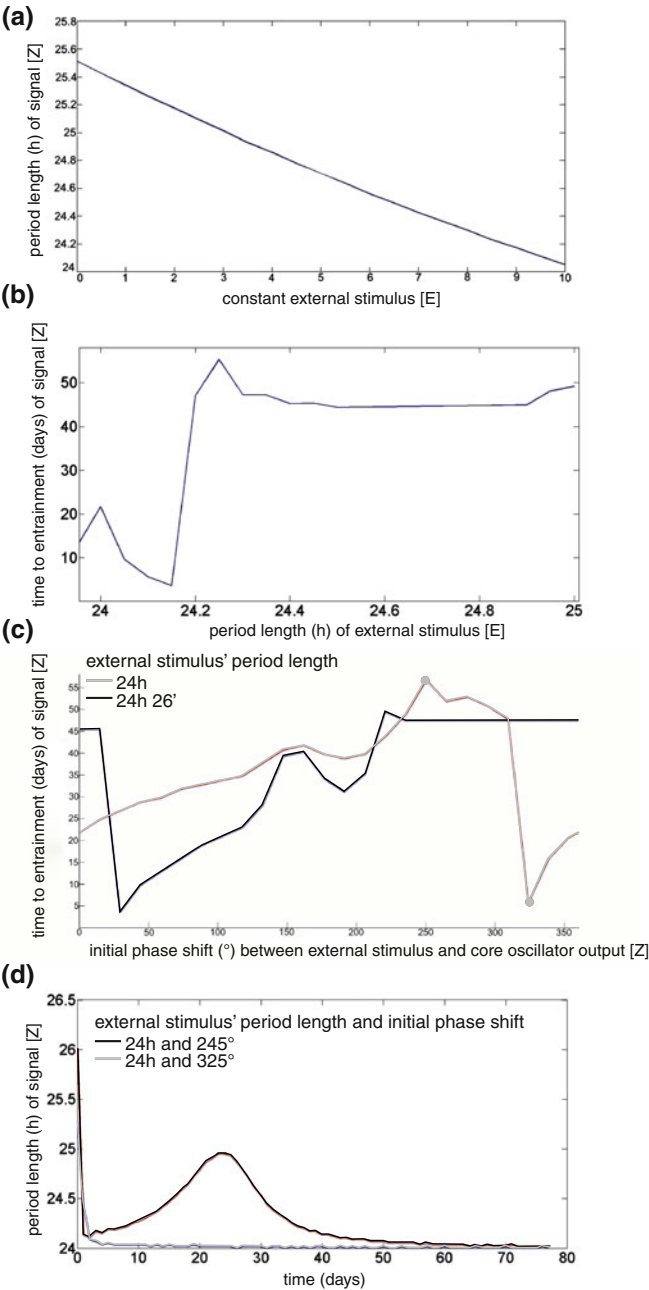


**Fig. 5.9** Bode plot derived from the low-pass filter in Fig. 5.8. Slope:  $\approx -100$  dB per frequency magnitude; cutoff frequency:  $\approx 1.04 \cdot 10^{-5} \text{ s}^{-1}$ . This value coincides with a period length of approx. 1.13 days



**Fig. 5.10** Chemical model of a PLL-based frequency control system comprising modules of a controllable core oscillator, a signal comparator, and a low-pass filter specified by underlying reaction kinetics or alternatively by corresponding transfer functions or characteristic curves. Modules are interconnected via shared molecular species

Finally, we are going to find out the ability of our control-system model to cope with both, different frequency *and* different initial phase between external stimulus  $L_I(E)$  and core oscillator output  $L_I(Z)$ . Figure 5.11c picks two constant frequencies for  $L_I(E)$ , one with a period length of 24h26' and a second course with a period length of exactly 24h. For both courses, the time to entrainment had been observed subject to initial phase shifts between 0 and 360°, which corresponds to a full period of the core oscillator's output  $L_I(Z)$ . It turns out that both courses reach specific



**Fig. 5.11** Simulation scenarios exploring the behaviour of our PLL-based control-system model. **a** Period lengths of sustained oscillations subject to constant external stimulus, **b** Time to entrainment to external stimulus whose frequency initially differs from those of core oscillator, **c** Ability of our control system to cope with both, different frequency and different initial phase between external stimulus and core oscillator, **d** Comparison of best case and worst case from entrainment run to 24 h subsumed in C

minima and maxima. Its global maximum (at approx.  $245^\circ$ ) and its global minimum (at approx.  $325^\circ$ ) come along with a detailed consideration of period length's variation over time towards entrainment in Fig. 5.11d.

## 5.4 Cell Signalling and Gene Regulatory Networks: Logic Circuits in Chronobiological Information Processing

Each biological cell is equipped with sophisticated mechanisms, which enable an appropriate response to manifold external and internal stimuli. Able to affect a cell at arbitrary points in time, these stimuli mainly comprise complex molecules with specific surface structures. In the same way a key fits into a lock in order to open the chamber behind, a stimulus binds at its receptor located within the outer membrane of the cell. Afterwards, the receptor initiates a sequence of intracellular reactions which in turn finally assemble a so-called transcription factor, another complex molecule composed from proteins and ligands. Production of transcription factors might include connection, weighting, and evaluation of molecular signals induced by different receptors. The term *cell signalling* subsumes the entirety of reactions available in this context. Having few copies of a transcription factor at hand, it can enter the cell nucleus controlling the expression of a specific gene. In most cases, several hundred up to some thousand individual copies of a transcription factor are enough to this end. Due to their spatial surface structure, transcription factors precisely bind at specific locations of the genomic DNA. This marks the starting point for gene expression. In concert with auxiliary substances, the adjacent coding DNA region (exon) is identified. Afterwards, the exon is transcribed into precursor-messenger RNA, which subsequently undergoes splicing or alternative splicing. Finally, the mature messenger RNA acts as template for translation into the resulting protein which represents the cellular response to the stimulus. Transcription factors in general can activate gene expression, but they are also able to inhibit gene expressions in the presence or absence of further messengers. Thus, the term *gene regulation* organised by gene regulatory networks came into use. A human cell for instance contains approx. 23,000 genes whose information enables the production of about 500,000 chemically distinct proteins, each of them combined from one or more coding regions [55]. While processes involved in cell signalling usually run within a minute-scale, the expression of a gene might consume several hours in total.

Understanding the time-dependent manner of cell signalling and gene regulation is a crucial aspect in chronobiological research towards achievement of fine-grained predictive models. Particularly, oscillatory signals revealed by a circadian clock act as triggers for manifold downstream processes. In human bodies, the release of serotonin in the morning as well as the segregation of melatonin in the evening for balanced daily variation of alertness and tiredness gives a typical example for periodical activities in cell signalling subject to the time of the day. Moreover, maintenance and optimisation of further much more complex processes of life require a more or

less concerted interplay in evaluation and compilation of numerous distinct signals. In this Section, we adopt a reliable cell signalling mechanism for construction and exploration of logic circuits by gene regulatory networks within an in-vivo application study. It results in providing a bistable molecular toggle switch. Membrane systems  $\Pi_{\text{CSM}}$  for cell signalling modules turn out to be appropriate formal representations of the underlying reaction systems due to their ability to cope with assembly of sub-molecular units towards complex structures. Interestingly, the consecutive change of molecular structures while passing a signalling cascade [7] exhibits a close similarity to a passage of distinct states known from finite automata in computer science.

### 5.4.1 The General Principle of Cell Signalling in vivo

Opposed to engineered networks (e.g. electronic circuits) whose topologies can be easily traced, biochemical network connections on their own are invisible. The circuitry of these natural networks is exclusively identified through interactions between their substrates. Biochemical reaction networks, found in pro- and eukaryotic cells, permit processes from which higher-level properties of life like behavioural patterns might emerge. Despite their high degree of complexity and interdependency, intracellular reaction networks are hierarchically arranged in modular structures of astonishing order. Related work in the context of P systems includes Refs. [6, 13, 30, 56].

A strong division of tasks, predefined transduction pathways as well as an efficient share of resources characterise biochemical networks. Mainly based on proteins as information carrier with high variability in molecular structure, the entirety of interconnected reaction processes constitutes the function of a cell and its subunits. Three essential types of biochemical networks in vivo can be distinguished: metabolic, cell signalling (CSN), and gene regulatory (GRN) networks [1]. Metabolism consists of coupled enzymatically catalysed reactions close to a minimum level of free energy. Corresponding networks mainly provide conserving and sustentative functions for the organism commonly operating to reach a steady state. In contrast, CSNs perform information processing by response to external or internal signals in concert with GRNs that control protein synthesis based on genetically conserved patterns. Slight malfunctions or perturbations within these fine-grained and sensitive network structures can have life-threatening consequences.

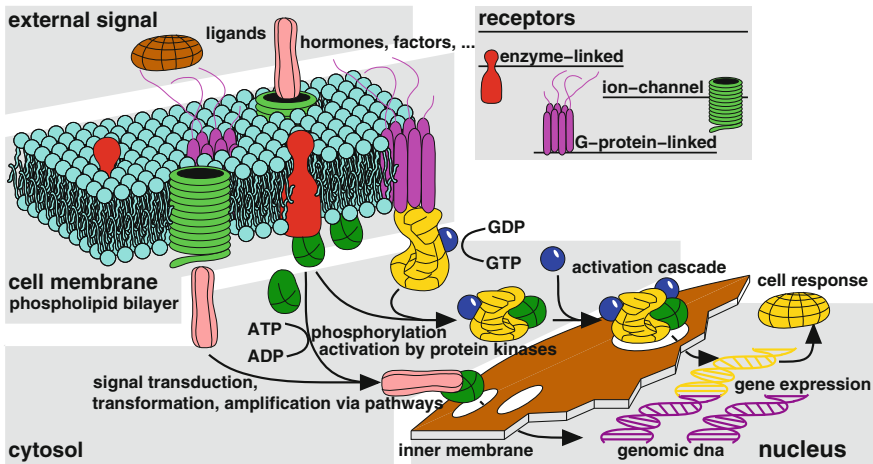
Proteins form central functional elements of the cell. They incorporate enzymes, factors, receptors, messengers, and subsidiary substances of which the cell is composed. Therefore, CSNs and GRNs, as central control systems for protein synthesis are of crucial impetus for both, maintenance and regeneration of organisms in concert with a certain ability to reply to environmental changes. In cell signalling, here exemplified by eukaryotic cells [38], three main steps can be identified (Fig. 5.12):

1. Signal reception: External signals arrive from other cells, from the environment, or from the cell's own feedback loops. These stimuli are encoded either by proteins or



hormones, auxiliary substances (like ions and ligands), or by physical conditions (e.g., light). They reach specific receptors embedded in the outer cell membrane. Ion channels transmit the signal by transporting substances into the inner cell; whereas enzyme-linked and G-protein-linked receptors transmit the signal simply by changing their conformation.

2. **Signal transduction:** Messenger proteins, originally bound to these receptors at the inner membrane face, are then emitted into the cytosol. Here, they initiate activation cascades for further signal transduction, evaluation, combination, and amplification. Activation of enzyme messengers mainly occurs by stepwise addition of phosphates from adenosinetriphosphate (ATP) to specific binding sites of messenger proteins. Alternatively, G-proteins bind to guanosinediphosphate (GDP) and guanosinetriphosphate (GTP). These processes can be accompanied by forming specific protein complexes.
3. **Cell response:** The resulting biomolecules then enter the nucleus where they can effect a specific gene expression controlled by a GRN, thus producing the cell response to the primary signal. The intensity of gene expression is determined by transcription factors. They act as promoters or repressors controlling the amount of messenger RNA transcribed from genomic DNA. Subsequent translations lead to the final protein. Typical biochemical networks can contain interactions between several hundred proteins including intermediate states and complexes.

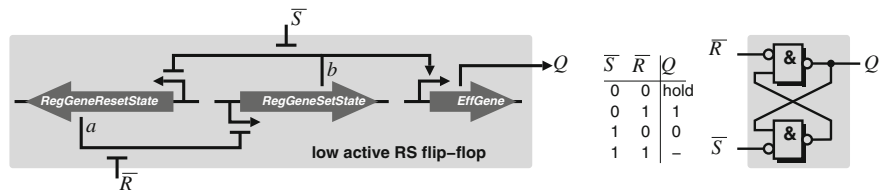


**Fig. 5.12** Biological principle of signalling in eukaryotic cells: from arriving stimuli to specific cell response

### 5.4.2 Modelling a Bistable Toggle Switch by a Gene Regulatory Network with Two Feedback Loops

A gene regulatory network can induce a bistable switching behaviour in a way to act as a simple RS flip-flop (bit storage unit with Set and Reset input) for information processing. Here, we consider two distinct input signals responsible for setting as well as resetting the flip-flop. The concentrations of two dedicated transcription factors over time are responsible for this task. For instance, they might be supplied by an upstream circadian clock output. In this scenario, the almost sinusoidal temporal course of the clock output becomes converted into a more plated signal course reflecting a more or less bistable oscillatory behaviour. Here, the waveform resembles an almost rectangular shape similar to a binary clock signal. Plated oscillations combine the advantage of fast toggling with the ability for a balanced or weightable ratio between high-level and low-level signal values. The fast raise and fall of the signal value is easy to detect for subsequent processing units. In addition, the functionality of a flip-flop downstream to a circadian clock can also help an organism to compare and to evaluate different signals, each of which contributing partially to a complex behavioural pattern. For example, the release of serotonin in human bodies to support alertness is also influenced by the adrenaline level in conjunction with the circadian clock signal (and even by further factors). A profound mechanism for evaluation among the variety of individual signals also takes into account temporarily memorised information about relevant signal values in the past. A reliable chemical flip-flop is an essential tool for this purpose. There is evidence for naturally evolved chemical flip-flops emerged in several forms. In this Section we introduce a simple example getting by with three genes illustrated in Fig. 5.13 adopted from Ref. [30].

The chemical flip-flop depicted in Fig. 5.13 operates by a mutual inhibition of gene expression, which either reproduces the set state or the reset state. Assuming the set state gene is active, its gene product  $b$  represses the synthesis of the reset state gene. In parallel,  $b$  activates the effector gene whose product  $Q$  represents the flip-flop output. The set state scenario remains stable until input  $\bar{S}$  constitutes a high concentration



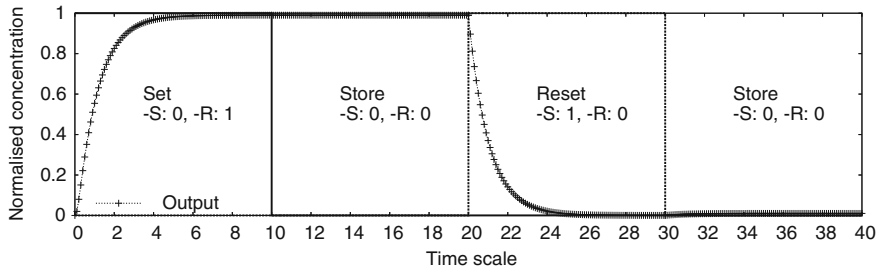
**Fig. 5.13** Scheme of an RS flip-flop composed by a gene regulatory network with mutually inhibiting feedbacks. High concentration of transcription factor  $\bar{S}$  sets the flip-flop to an output  $Q$  of logical zero while a high concentration of transcription factor  $\bar{R}$  leads to a high-level output  $Q$ . Low concentrations at both inputs  $\bar{S}$  and  $\bar{R}$  store the previously set output. Simultaneous setting and resetting is forbidden producing a medium concentration level of  $Q$  away from a clear assignment to logical 1 or 0

level and  $\bar{R}$  a low one. Now, a toggling process occurs. Transcription factor  $b$  is no longer transduced within the circuit. In consequence, the reset state gene starts expressing its product  $a$ , which in turn additionally suppresses production of  $b$ .  $Q$  cannot regenerate anymore, its concentration diminishes towards a low concentration close to zero.

The chemical RS flip-flop from Fig. 5.13 can be described by a P system of type  $\Pi_{\text{CSM}}$ . For specification of the dynamical behaviour of gene expression we apply Hill kinetics in its standard form [14] in combination with mass-action degradation of involved transcription factors:

$$\begin{aligned}
 \Pi_{rsff} &= (V, V', R_1, \dots, R_6, f_1, \dots, f_6, A, C, \Delta\tau) \\
 V &= \{\bar{R}, \bar{S}, Q, a, b\} \\
 V' &= \emptyset \\
 R_1 &= \bar{S} + b \longrightarrow a \\
 R_2 &= \bar{R} + a \longrightarrow b \\
 R_3 &= b \longrightarrow Q \\
 R_4 &= a \longrightarrow \emptyset \\
 R_5 &= b \longrightarrow \emptyset \\
 R_6 &= Q \longrightarrow \emptyset \\
 f_1(L_t) &= 1 - \frac{L_t(b)^m}{L_t(b)^m + \Theta_b} \cdot \left(1 - \frac{L_t(\bar{S})^m}{L_t(\bar{S})^m + \Theta_{\bar{S}}}\right) \\
 f_2(L_t) &= 1 - \frac{L_t(a)^m}{L_t(a)^m + \Theta_a} \cdot \left(1 - \frac{L_t(\bar{R})^m}{L_t(\bar{R})^m + \Theta_{\bar{R}}}\right) \\
 f_3(L_t) &= \frac{L_t(b)^m}{L_t(b)^m + \Theta_b} \cdot \left(1 - \frac{L_t(\bar{S})^m}{L_t(\bar{S})^m + \Theta_{\bar{S}}}\right) \\
 f_4(L_t) &= k_a \cdot L_t(a) \\
 f_5(L_t) &= k_b \cdot L_t(b) \\
 f_6(L_t) &= k_Q \cdot L_t(Q)
 \end{aligned}$$

Figure 5.14 shows a simulation study demonstrating the dynamical behaviour of the flip-flop  $\Pi_{rsff}$  by using a time scale of hours. After 10h,  $\bar{R}$ 's concentration is set to zero while  $\bar{S}$  constitutes a high concentration level of 1 between 20 and 30h at time scale. We observe the expected behaviour of a flip-flop able to store its set or reset state. Each toggling causes a certain latency up to release of valid output signal levels.



**Fig. 5.14** Temporal concentration course of flip-flop output  $Q$  subject to subsequent setting and resetting flanked by intermediate storing phases.  $\Pi_{rsff}$  parameter setting:  $m = 2$ ,  $\Theta_a = \Theta_b = \Theta_{\bar{S}} = \Theta_{\bar{R}} = 0.1$ ,  $k_a = k_b = k_Q = 1$ ,  $A = L_0 = \{(\bar{R}, 1), (\bar{S}, 0), (a, 0), (b, 0), (Q, 0)\}$ ,  $C = 20$ ,  $\Delta\tau = 0.05$

### 5.4.3 In Vivo Implementation of a Bistable Toggle Switch

We demonstrate the practicability of the RS flip-flop by an experimental study in vivo. Following the pioneering implementation of a bistable toggle switch [18], we could confirm its function in a previous study [27]. Two extensions were made: Firstly, the effects of isopropyl- $\beta$ D-thiogalactopyranoside (IPTG) and N-acyl homoserine lactone (AHL) as appropriate intercellular inducers  $\bar{S}$  and  $\bar{R}$  for flip-flop setting and resetting were shown. Secondly, flow cytometry was used to quantitatively measure protein concentrations within the flip-flop implementation. We give a brief overview of experimental setup and results.

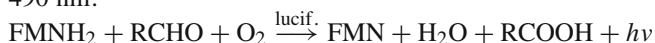
#### 5.4.3.1 Biological Principles and Prerequisites

##### Quorum Sensing and Autoinduction via AHL

In quorum sensing, bacterial species regulate gene expression based on cell-population density [37, 45]. An alteration in gene expression occurs when an intercellular signalling molecule termed autoinducer, produced and released by the bacterial cells reaches a critical concentration. Termed as quorum sensing or autoinduction, this fluctuation in autoinducer concentration is a function of bacterial cell-population density. *Vibrio fischeri*, a well studied bacterium, colonises the light organs of a variety of marine fishes and squids, where it occurs at very high densities ( $10^{10} \frac{\text{cells}}{\text{ml}}$ ) and produces light [57]. The two genes essential for cell density regulation of luminescence are *luxI*, which codes for an autoinducer synthase and *luxR*, which codes for an autoinducer-dependent activator of the luminescence genes. The *luxR* and *luxI* genes are adjacent and divergently transcribed, and *luxI* is the first of seven genes in the luminescence or *lux* operon. *LuxI*-type proteins direct AHL synthesis while *LuxR*-type proteins function as transcriptional regulators that are capable of binding AHL signal molecules. Once formed, *LuxR*-AHL complexes bind to target promoters of quorum-regulated genes. Quorum sensing is by now known to be widespread among both Gram-positive and Gram-negative bacteria.

### **Bioluminescence in *Vibrio fischeri***

Bioluminescence in general is defined as an enzyme catalysed chemical reaction in which the energy released is used to produce an intermediate or product in an electronically excited state, which then emits a photon. It differs from fluorescence or phosphorescence as it is not depended on light absorbed. The mechanism for gene expression and the structure of the polycistronic message of the lux structural genes in *Vibrio fischeri* have been thoroughly characterised [25]. Briefly, there are two substrates, luciferin, which is a reduced flavin mononucleotide (FMNH<sub>2</sub>), and a long chain (7–16 carbons) fatty aldehyde (RCHO). An external reductant acts via flavin mono-oxygenase oxidoreductase to catalyse the reduction of FMN to FMNH<sub>2</sub>, which binds to the enzyme and reacts with O<sub>2</sub> to form a 4a-peroxy-flavin intermediate. This complex oxidises the aldehyde to form the corresponding acid (RCOOH) and a highly stable luciferase-hydroxyflavin intermediate in its excited state, which decays slowly to its ground state emitting blue-green light  $h\nu$  with a maximum intensity at about 490 nm:



### **Transcription Control by LacR and $\lambda$ CI Repressor Proteins**

*Escherichia coli* cells repress the expression of the lac operon when glucose is abundant in the growth medium. Only when the glucose level is low and the lactose level is high, the operon is fully expressed. The Lac repressor LacR is a 360 residue long protein that merges to a homotetramer. It contains a helix-turn-helix (HTH) motif through which it interacts with DNA. This interaction represses transcription by impeding association with RNA polymerase and represents an example of combinatorial control widely seen in prokaryotes and eukaryotes. The CI repressor of bacteriophage lambda is the key regulator in lambda's bistable genetic switch that underlies the phage's ability to efficiently use its two modes of development.

### **Flow Cytometry**

Flow cytometry refers to the technique where microscopic particles are counted and examined as they pass in a hydrodynamically focused fluid stream through a measuring point surrounded by an array of detectors. Previously, we performed flow cytometry analyses by using a BD LSRII flow cytometer equipped with 405, 488 and 633 nm lasers. 488 nm laser was used for *gfp* and yellow fluorescent protein (*yfp*) quantification.

#### **5.4.3.2 Experimental Setup and Implementation**

We have shown that an in vivo system [27] can potentially be used to mimic an RS flip-flop and have quantified its performance using flow cytometry. The presence or absence of the inducers IPTG or AHL in combination with temperature shift acts as an input signal, see Fig. 5.15. The toggle switch comprising of structural genes for reporter/output proteins connected to promoter regions that are regulated by input signals is visualised as an RS flip-flop. This design endows cells with two distinct phenotypic states where the  $\lambda$ CI activity is high and the expression of lacI is low

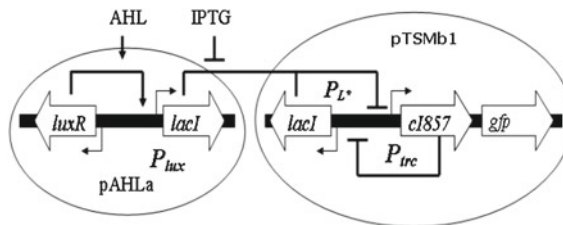
(referred to as high or 1 state), or the activity of LacR is high and the expression of  $\lambda$ CI is low (referred to as low or 0 state). The green fluorescent protein *gfp* is expressed only in the high  $\lambda$ CI/low LacR state.

### 5.4.3.3 Results and Discussion

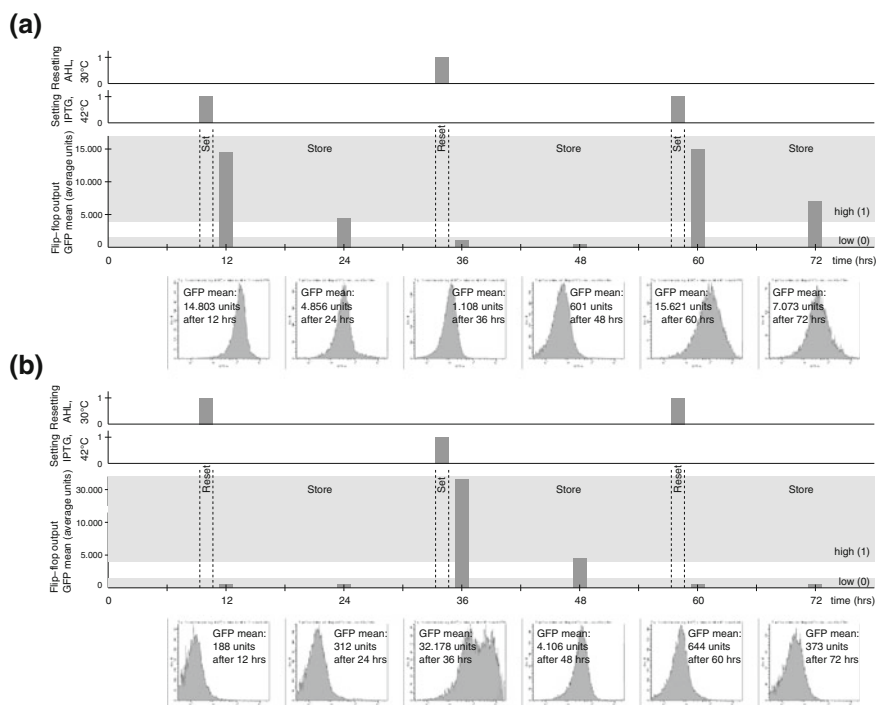
For co-relational purposes, all experiments were conducted with both BL21 and Top10 strains of *Escherichia coli*. The concentration of IPTG used in all the experiments was 2 mM and that of AHL was 1  $\mu$ M. Experiments conducted without the use of inducers lead to an unreliable shift of the states, indicating the use of inducers in a tightly, mutually regulated circuit. Further experiments conducted to understand the switching dynamics of the circuit revealed that in the current scenario it was easier to switch from a high to a low state than vice versa. This discrepancy in switching behaviour is attributed to the different modes of elimination of LacR and  $\lambda$ CI repressor proteins. While switching from low to high state, the repression due to IPTG-bound Lac repressor needs to be overcome by cell growth. Switching from high to low state is effected by immediate thermal degradation of the temperature-sensitive  $\lambda$ CI. Experiments were also conducted to test the sustainability of states. The plug and play property of the circuit was examined by employing *yfp* as the reporter gene instead of *gfp*. As shown in Fig. 5.16, the circuit could mimic an RS flip-flop. A massive parallelism by the use of large quantities of cells can compensate for the slow speed of switching. Further tests are to be performed to confirm this hypothesis.

## 5.5 Spatial Rule-Based Simulator Software SRSim at a Glance

Molecular dynamics opens a promising framework towards modelling and simulation of reaction systems whose substrates can spatially compose or decompose from submolecular units following the intention of membrane systems  $\Pi_{\text{CSM}}$  for cell signalling modules. In this line, the aim of the resulting *SRSim* software package is coping with reaction networks that are combinatorially complex as well as spatially



**Fig. 5.15** A schematic diagram of an AHL biosensor module interfaced with the genetic toggle switch adapted from Ref. [27]. The transgenic artificial GRN consists of a bistable genetic toggle switch [18] which is linked to genes from the *lux* operon of the quorum sensing signalling pathway of *Vibrio fischeri*



**Fig. 5.16** Inducer-dependent switching. Repeated activation and deactivation of the toggle switch based on inducers and temperature. Temperature was switched every 24h. Cells were incubated with inducers for 12h, followed by growth for 12h without inducers, initially kept at 30 (a) and 42 °C (b). The cells successfully switched states three times. Figure taken from Ref. [30]

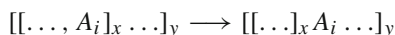
inhomogeneous. It is constructed from the molecular dynamics simulator LAMMPS [54] and a set of extensions for modeling rule-based reaction systems. On the one hand, there is a potentially combinatorial explosion of necessary species and reactions that occurs when complex biomolecules are allowed to interact, e.g. by polymerisation or phosphorylation processes. On the other hand, diffusion over longer distances in the cell as well as the geometric structures of sophisticated macromolecules can further influence the dynamical behaviour of a system. Addressing these demands, the SRSim simulation system features a stochastic, particle based, spatial simulation of Brownian Dynamics in three dimensions of a rule-based reaction system. This Section gives a brief overview of SRSim based on [23, 24].

In our approach, elementary molecules consist of a set of components or domains which can be modified or bound to the components of other molecules. Components, sites, binding sites and domains are used synonymously in this context. The resulting complex species, formed by a connection of elementary molecules, are called molecule graphs. Instead of using reactions between explicit species now, the reactions are replaced by implicit reaction rules, which are applicable to a certain

subset of all possible complex molecular species. This subset is defined through an equivalence class given by a molecule graph pattern. Any complex molecule graph that contains the graph pattern as an isomorphic sub-pattern is included in the equivalence class. A pattern might, for example, describe a molecule of type *A* that has one free binding site and another binding site bound to another molecule of type *A*. Any other molecular species that incorporates this *A*–*A* dimer without blocking the necessary free binding site specified in the pattern is now part of this equivalence class. Any of these molecules can be addressed by a single reaction rule, as demonstrated in Fig. 5.17.

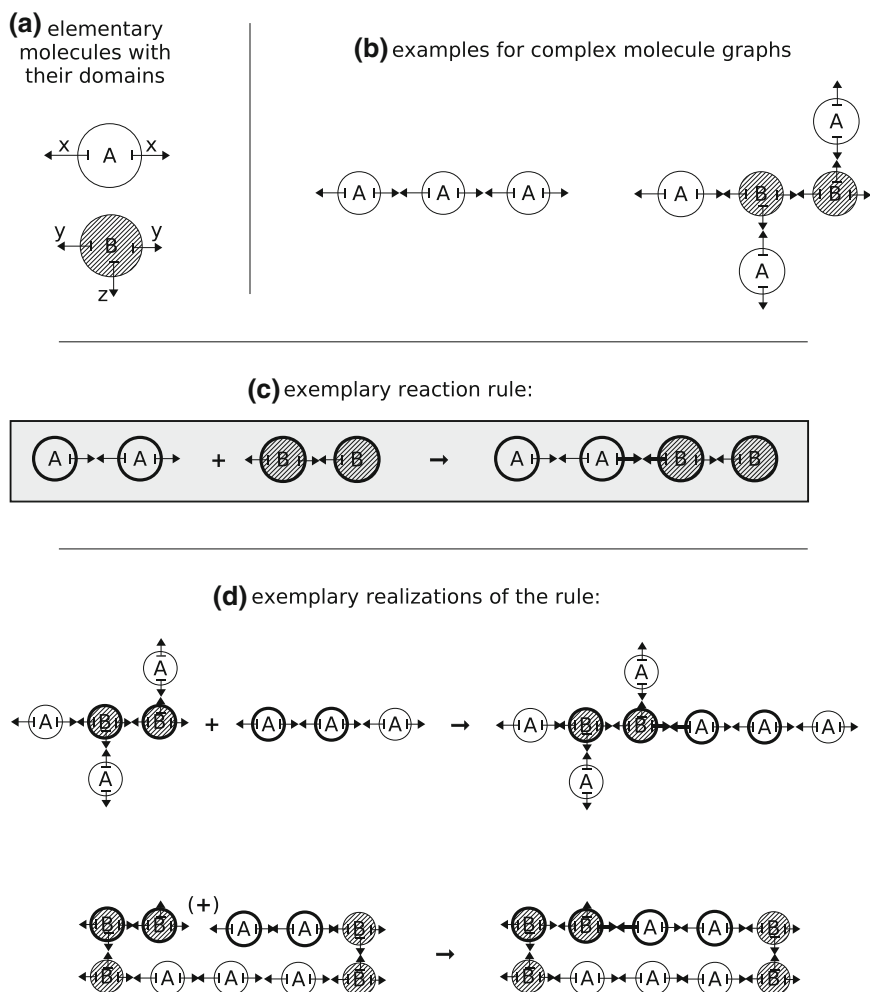
We are using individual agents for each elementary molecule in the simulation. Synonymously with the agents, we are using the terms particles and elementary molecules here. The spatial simulation is carried out as an extension to the molecular dynamics simulator LAMMPS [54]. Each particle is represented by its position, its velocity, its species and the state of its components. The particles diffuse through the reactor and can push away other molecules if they come too close. When two molecules approach one another, a bimolecular reaction can happen between them, if they are fitting to a reaction pattern specified in the reaction rules and if the geometric constraints are met. If a reaction binds two elementary molecules together, bond forces are applied and their diffusion through the reactor is coupled, forming a complex molecule graph. Monomolecular reactions can be used to spontaneously break bonds in molecule graphs or to modify the component states of a molecule. More conventional reactions can also be used to completely exchange one molecule for another. The inclusion of spatial aspects in the rule-based reaction systems results in an expressive simulation system for moderately sized systems. Up to a magnitude of 100,000 particles the simulation can still be run on a desktop system for about  $10^6$  timesteps in some hours of computing time. The rule-based reaction system is specified in the BioNetGen Language [34] (BNGL).<sup>1</sup>

SRSim can be interpreted as a P system with membranes composed from assembled individual molecules. Reactions are constrained by spatial configurations and geometries which may include molecules forming parts of membranes. So far, membranes can be defined as static force fields or can emerge (e.g. like lipid layers formations [63]). Although this way of implicit membrane definition by molecular assembly is computationally demanding, it provides a high level of granularity in dealing with issues of reception of molecular passage through membrane structures. To do so, geometric properties like a form (e.g. sphere), a location, a size and a velocity are added to a membrane, so that it can have an effect on and can be affected by spatial heterogeneities. For example, a reaction to make a molecule of species *A* leaving a membrane *x*



<sup>1</sup> See the BioNetGen Documentation. A BioNetGen tutorial can be found online at: [http://bionetgen.org/index.php/BioNetGen\\_Tutorial](http://bionetgen.org/index.php/BioNetGen_Tutorial)





**Fig. 5.17** Exemplary rule-based system, taken from [23]. Two elementary molecule types ( $A$ ,  $B$ ) with their subdomains (or components) are displayed **(a)**. Each component can be bound to another component or be modified, e.g. denoting a phosphorylation or a conformational change. Site names need not be unique and hence a wide spectrum of possibilities for the system's specification is offered. Multiple elementary molecules can be connected at their components to form complex molecule graphs **(b)**. Reaction rules, as the binding reaction **(c)**, are specified by using patterns graphs (or reactant patterns). A reactant pattern fits to a molecule graph, if it is contained as a subgraph in the molecule graph. Note that some components are missing in the reactant pattern's definition, which are then ignored in the matching process. Panel **(d)** shows two different instances of the reaction rule. In the upper realization, two independent molecule graphs are connected. For the lower example on the other hand, both of the rules' reactant patterns are found in a single connected molecule graph

might require an appropriate particle  $A_i$  to be situated close to the membrane, before it can exit. This notation closely follows the scheme of symport/antiport rules or active membranes in P systems. Other constraints follow easily, e.g. mean transition times from one membrane into another compartment that is situated some distance away.

SRSim as well as LAMMPS are released under the GNU Public License, so the sources can freely be downloaded<sup>2</sup> and modified. Especially the rule system that handles the rule-based reaction system is independent from the molecular dynamics simulator and could be plugged into a different spatial or non-spatial realisation of a rule-based simulation system.

## 5.6 Envisioning an Analysis of Membrane System's Static and Dynamical Behaviour by a Constrained-Based Approach

Within the domain of strictly continuous signals quantified by real numbers, modelling and analysis of oscillating behaviour has been well-studied [53]. Chemical reaction networks assumed to reside in a homogeneous environment give a typical example: Each species is represented by its concentration which is allowed to vary continuously over time. From the static network topology together with the stoichiometry of the reactions, a corresponding ordinary differential equation system (ODE) can be derived that specifies the reaction rates for each species [14]. Inclusion of parameterised kinetic laws accomplishes a mapping between species concentration and effective reaction rate. The resulting ODE can easily be tested for its capability of undamped oscillating species concentrations. To this end, the *eigenvalues* of the *Jacobian matrix* obtained from the ODE right hand side are sufficient [26]. *Limit cycles* indicate the oscillatory behaviour in detail. In case of sinusoidal or almost sinusoidal oscillatory waveforms, even properties of the entrainment behaviour can be obtained analytically, for instance by the Kuramoto method [39] which estimates the expected time to synchronisation subject to the range of relevant parameters.

The main advantage of analytical ODE-based methods unequivocally exploits the fact that essential characteristics and properties of a system under study can be directly derived from the underlying mathematical model without any need for a numerical simulation of its dynamical behaviour. This makes the evaluation and automated testing of candidate systems resulting from experimental data rather efficient. In contrast, there is currently a lack of corresponding methods within the field of membrane systems modelled in a discrete manner. Here, system properties mainly emerge from exhaustive simulation studies. Conduction of those studies still requires an extensive amount of human resources. Particularly in case of involved active membranes, compartmental plasticity, and dynamical structures, a toolbox for automated analysis would be helpful. In an ongoing project, we intend to generate sustained oscillatory systems by artificial evolution in silico [40]. In this context, the

---

<sup>2</sup> [www.biosystemsanalysis.de](http://www.biosystemsanalysis.de)

fitness evaluation should answer the question whether the system candidates are able to oscillate endogenously or not and how the periodicity can be controlled. This task is expected to be done by a piece of software [23].

There are different oscillatory scenarios in biological systems even beyond circadian clocks. On the one hand, periodicity might also be reflected in temporal changes of the compartmental structure. On the other hand, signalling molecules are often available in low concentrations ranging from single molecules to several thousand copies. Both aforementioned scenarios have in common to prevent pure ODE-based modelling techniques due to the discrete manner of involved key entities. Motivated by the need for an appropriate toolbox covering description, simulation, and analysis of discontinuously considered biological reaction processes, we plan to extend the concept of spatiotemporal membrane systems with kinetic laws [16, 29] towards an underlying evaluation strategy able to explore the nature of undamped oscillations beyond variations of species concentration. A simple approach combines numerical simulation with a backtracking mechanism. Here, the trace of configurations passed by a membrane system becomes recorded in a suitable way. By monitoring the overall configurations over time, a derivation tree is obtained that provides a comprehensive data pool for further analysis by automated backtracking. Sustained oscillations are expected to appear as recurring, but non-adjacent overall configurations along a path through the derivation tree. The practicability of simple backtracking for detection of oscillatory behaviour is limited to small or medium sized membrane systems due to the need of a fine-grained numerical simulation, which is time-consuming. Particularly for fitness evaluation in artificial evolution, numerous network candidates within a network population have to be analysed.

Exploitation of a purely *constrained-based* approach [35, 59] could be a promising clue in order to decide about dedicated properties of membrane systems without any need for numerical simulation of system's behaviour. In this context, a constraint defines a valid range of system's parameterisation which implies a corresponding qualitative behaviour to be obtained by reasoning from the formal system's description. An example reflecting the static behaviour is given by the decision whether or not a reaction system on its own is *mass-conserving*. This property exclusively follows from the reaction equations. In each equation, the set of products must be completely composed from the reactants without any loss or overrun. In case of the single reaction  $A + B \longrightarrow C$ , mass conservation implies that product  $C$  is composed from one molecule  $A$  and one molecule  $B$ . Different compositions of  $C$  prevent mass conservation. Let us assume, we have in addition a second reaction  $C \longrightarrow A$ . Now, the entire reaction system is not mass-conserving since there is no consistent representation of  $C$ 's composition assuring mass balance in total.

The topology of a reaction network can disclose some information about dynamical properties. A sustained oscillatory behaviour for instance requires at least one reaction cycle as a mandatory prerequisite. A second analysis stage figures out whether or not there is a reaction cycle organised in a feedback expressed by a side reaction which ends up in the cycle (inflow) or leaves the cycle (outflow). The third analysis stage takes into account stoichiometry in conjunction with kinetic laws of reactions assigned to a feedback loop. Here, the type of kinetics together

with stoichiometric factors indicate the oscillatory capability of the feedback loop under study. Finally, all feedback loops identified within the entire system undergo a check on possible interdependencies able to influence the overall oscillatory behaviour. Certainly, the sketch based on passing four stages gives just a raw impression on how a constrained-based approach could act in general. Future work intends to be directed on refinement and adaptation of constrained-based methods for analysis of membrane systems envisioning preparation of a beneficial toolbox.

## 5.7 Conclusions

Chronobiological research can benefit from a consistent membrane system-based framework  $\Pi_{\text{CSM}}$  able to combine dynamical molecular structures with appropriate reaction kinetics. Particularly in cell signalling and gene regulation, both characterised by a low number of complex molecules involved,  $\Pi_{\text{CSM}}$ 's descriptiveness and expressivity makes it an advantageous tool for modelling issues in systems biology. We have demonstrated the underlying framework by three application studies: The KaiABC core oscillator exemplifies a post-translational component of a circadian clock in cyanobacteria while circadian clocks in general can emulate frequency control by phase-locked loops. Finally, the third study sheds light on an in-vivo implementation of a bistable toggle switch whose mutually feedbacked gene regulatory loops mimic an RS flip-flop variant. The application studies came along with the simulation system SRSim which is freely available at [66]. Further information on the network models can be obtained from the download area at Ref. [67].

## References

1. B. Alberts, *Essential Cell Biology* (Garland Publishing, London, 2003)
2. U. Albrecht, *The Circadian Clock* (Springer Verlag, New York, 2010)
3. U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits* (Chapman and Hall, Boca Raton, 2006)
4. A.P. Arkin, Synthetic cell biology. *Curr. Opin. Biotechnol.* **12**(6), 638–644 (2011)
5. I.M. Axmann, S. Legewie, H. Herzog, A minimal circadian clock model. *Genome Inf.* **18**, 54–64 (2007)
6. N. Barbacari, A. Profir, C. Zelinschi, Gene regulatory network modelling by means of membrane systems, in *7th International Workshop on Membrane Computing WMC 2006*, ed. by H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa, vol. 4361 (LNCS, 2006), pp. 162–178
7. J. Behre, S. Schuster, Modeling signal transduction in enzyme cascades with the concept of elementary flux modes. *J. Comput. Biol.* **16**(6), 829–844 (2009)
8. B.W. Bequette, *Process Control: Modeling, Design, and Simulation* (Prentice Hall, Upper Saddle River, 2003)
9. F. Bernardini, V. Manca, Dynamical aspects of P systems. *BioSystems* **70**, 85–93 (2003)
10. R.E. Best, *Phase-Locked Loops: Design, Simulation, and Applications* (McGraw-Hill, New York, 2007)

11. M.L. Blinov, J.R. Faeder, B. Goldstein, W.S. Hlavacek, BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics* **20**, 3289–3292 (2004)
12. M.L. Blinov, J.R. Faeder, B. Goldstein, W.S. Hlavacek, A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. *BioSystems* **83**, 136–151 (2006)
13. N. Busi, C. Zandron, Computing with genetic gates, proteins and membranes, in *7th International Workshop on Membrane Computing WMC 2006*, ed. by H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa, vol. 4361 (LNCS, 2006), pp. 213–228
14. K.A. Connors, *Chemical Kinetics* (VCH Publishers, New York, 1990)
15. P. Dittrich, J. Ziegler, W. Banzhaf, Artificial chemistries—a review. *Artif. Life* **7**(3), 225–275 (2001)
16. F. Fontana, V. Manca, Discrete solutions to differential equations by metabolic P systems. *Theor. Comput. Sci.* **372**(2–3), 165–182 (2007)
17. P. Frisco, *Computing with Cells. Advances in Membrane Computing* (Oxford University Press, Oxford, 2009)
18. T.S. Gardner, C.R. Cantor, J.J. Collins, Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**, 339–342 (2000)
19. D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977)
20. D.T. Gillespie, A rigorous derivation of the chemical master equation. *Physica A* **188**(1–3), 404–425 (1992)
21. S.S. Golden, V.M. Cassone, A. LiWang, Shifting nanoscopic clock gears. *Nat. Struct. Mol. Biol.* **14**, 362–363 (2007)
22. B.C. Goodwin, Oscillatory behaviour in enzymatic control processes. *Adv. Enzyme Regul.* **3**, 425–438 (1965)
23. G. Gruenert, B. Ibrahim, T. Lenser, M. Lohel, T. Hinze, P. Dittrich, Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinform.* **11**, 307 (2010)
24. G. Gruenert, P. Dittrich, Using the SRSim software for spatial and rule-based modelling of combinatorially complex biochemical reaction systems, in *11th International Conference on Membrane Computing CMC 2010*, ed. by M. Gheorghe, T. Hinze, G. Păun, G. Rozenberg, A. Salomaa, vol. 6501 (LNCS, 2010), pp. 240–256
25. J. Hastings, K.H. Nealson, Bacterial bioluminescence. *Ann. Rev. Microbiol.* **31**, 549–595 (1977)
26. B.A. Hawkins, H.V. Cornell, *Theoretical Approaches to Biological Control* (Cambridge University Press, Cambridge, 1999)
27. S. Hayat, K. Ostermann, L. Brusch, W. Pompe, G. Rödel, Towards in vivo computing: quantitative analysis of an artificial gene regulatory network behaving as an RS flip-flop, in *1st International Conference on Bio Inspired Models of Network, Information and Computing Systems BIONETICS 06*, ed. by T. Suda, C. Tschudin (ACM, 2006)
28. R. Heinrich, S. Schuster, *The Regulation of Cellular Systems* (Springer-Verlag, 2006)
29. T. Hinze, T. Lenser, P. Dittrich, A protein substructure based P system for description and analysis of cell signalling networks, in *7th International Workshop on Membrane Computing WMC 2006*, ed. by H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa, vol. 4361 (LNCS, 2006), pp. 409–423
30. T. Hinze, S. Hayat, T. Lenser, N. Matsumaru, P. Dittrich, Hill kinetics meets P systems, in *8th International Workshop on Membrane Computing WMC 2007*, ed. by G. Eleftherakis, P. Kefalas, G. Păun, G. Rozenberg, A. Salomaa, vol. 4860 (LNCS, 2007), pp. 320–335
31. T. Hinze, R. Fassler, T. Lenser, P. Dittrich, Register machine computations on binary numbers by oscillating and catalytic chemical reactions modelled using mass-action kinetics. *Int. J. Found. Comput. Sci.* **20**(3), 411–426 (2009)
32. T. Hinze, T. Lenser, G. Escuela, I. Heiland, S. Schuster, Modelling signalling networks with incomplete information about protein activation states: a P system framework of the KaiABC oscillator, in *10th International Workshop on Membrane Computing WMC 2009*, ed. by G.

- Păun, M.J. Perez-Jimenez, A. Riscos-Nunez, G. Rozenberg, A. Salomaa, vol. 5957 (LNCS, 2010), pp. 316–334
33. T. Hinze, C. Bodenstein, B. Schau, I. Heiland, S. Schuster, Chemical analog computers for clock frequency control based on P modules, in *12th International Conference on Membrane Computing CMC 2011*, ed. by M. Gheorghe, G. Păun, G. Rozenberg, A. Salomaa, S. Verlan, vol. 7184 (LNCS, 2012), pp. 182–202
  34. W.S. Hlavacek, J.R. Faeder, M.L. Blinov, R.G. Posner, M. Hucka, W. Fontana, Rules for modeling signal-transduction systems. *Sci. STKE* **344**, re6 (2006)
  35. P. Hofstedt, *Multiparadigm Constraint Programming Languages. Series Cognitive Technologies* (Springer, New York, 2011)
  36. W.L. Koukkari, R.B. Sothorn, *Introducing Biological Rhythms* (Springer Verlag, Berlin, 2006)
  37. N. Krasnogor, M. Gheorghe, G. Terrazas, S. Diggle, P. Williams, M. Camara, An appealing computational mechanism drawn from bacterial quorum sensing. *Bull. EATCS* **85**, 135–148 (2005)
  38. G. Krauss, *Biochemistry of Signal Transduction and Regulation* (Wiley-VCH, Weinheim, 2003)
  39. Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulences* (Springer Verlag, Berlin, 1984)
  40. T. Lenser, T. Hinze, B. Ibrahim, P. Dittrich, Towards evolutionary network reconstruction tools for systems biology, in *5th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics EvoBIO2007*, ed. by E. Marchiori, J.H. Moore, J.C. Rajapakse, vol. 4447 (LNCS, 2007), pp. 132–142
  41. R.D. Lewis, Control systems models for the circadian clock of the New Zealand Weta *Hemideina thoracia*. *J. Biol. Rhythms* **14**, 480–485 (1999)
  42. M.O. Magnasco, Chemical kinetics is turing universal. *Phys. Rev. Lett.* **78**(6), 1190–1193 (1997)
  43. V. Manca, L. Bianco, F. Fontana, Evolution and oscillation in P systems: applications to biological phenomena, in *5th International Workshop on Membrane Computing WMC 2004*, vol. 3365 (LNCS, 2005), pp. 63–84
  44. M. Marhl, M. Perc, S. Schuster, Selective regulation of cellular processes via protein cascades acting as band-pass filters for time-limited oscillations. *FEBS Lett.* **579**(25), 5461–5465 (2005)
  45. M.B. Miller, B.L. Bassler, Quorum sensing in bacteria. *Ann. Rev. Microbiol.* **55**, 165–199 (2001)
  46. T. Mori, D.R. Williams, M.O. Byrne, X. Qin, M. Egli, H.S. Mchaourab, P.L. Stewart, C.H. Johnson, Elucidating the ticking of an in vitro circadian clockwork. *PLoS Biol.* **5**(4), 841–853 (2007)
  47. M. Nakajima, K. Imai, H. Ito, T. Nishiwaki, Y. Murayama, Reconstitution of circadian oscillation of cyanobacterial KaiC phosphorylation in vitro. *Science* **308**, 414–415 (2005)
  48. J. O'Neill, G. Ooijen, L.E. Dixon, C. Troein, F. Corellou, F.Y. Bouget, A.B. Reddy, A.J. Millar, Circadian rhythms persist without transcription in a eukaryote. *Nature* **469**, 554–558 (2011)
  49. T. Nishiwaki, Y. Satomi, Y. Kitayama, K. Terauchi, R. Kiyohara, T. Takao, T. Kondo, A sequential program of dual phosphorylation of KaiC as a basis for circadian rhythm in cyanobacteria. *EMBO J.* **26**, 4029–4037 (2007)
  50. D.A. Paranpje, V.K. Sharma, Evolution of temporal order in living organisms. *J. Circadian Rhythms* **3**, 7 (2007)
  51. G. Păun, Computing with membranes. *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000)
  52. G. Păun, *Membrane Computing* (Springer Verlag, Berlin, 2002)
  53. T. Pavlidis, *Biological Oscillators: Their Mathematical Analysis* (Academic Press, New York, 1974)
  54. S.J. Plimpton, Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.* **117**, 1–19 (1995)
  55. F. Ponten, K. Jirstrom, M. Uhlen, The human protein atlas: a tool for pathology. *J. Pathol.* **216**(4), 387–393 (2008)
  56. F.J. Romero-Campero, M.J. Perez-Jimenez, Modelling gene expression control using P systems: the Lac Operon, a case study. *Biosystems* **91**(3), 438–457 (2008)

57. F.J. Romero-Campero, M.J. Perez-Jimenez, A model of the quorum sensing system in *Vibrio fischeri* using P systems. *Artif. Life* **14**(1), 95–109 (2008)
58. E. Rosato, *Circadian Rhythms: Methods and Protocols* (Springer-Verlag, New York, 2007)
59. F. Rossi, P. van Beek, T. Walsh, *Handbook of Constraint Programming* (Elsevier Science, Amsterdam, 2006)
60. M.R. Roussel, D. Gonze, A. Goldbeter, Modeling the differential fitness of cyanobacterial strains whose circadian oscillators have different free-running periods. *J. Theor. Biol.* **205**(2), 321–340 (2000)
61. P. Ruoff, M. Vinsjevsk, C. Monnerjahn, L. Rensing, The Goodwin oscillator: on the importance of degradation reactions in the circadian clock. *J. Biol. Rhythms* **14**(6), 469–479 (1999)
62. M. Samoilov, A. Arkin, J. Ross, Signal processing by simple chemical systems. *J. Phys. Chem. A* **106**(43), 10205–10221 (2002)
63. J. Smaldon, N. Krasnogor, C. Alexander, M. Gheorghe, Liposome logic, in *11th Annual Conference on Genetic and Evolutionary Computation GECCO 2009*, ed. by F. Rothlauf (ACM, 2009), pp. 161–168
64. J. Tomita, M. Nakajima, T. Kondo, H. Iwasaki, No transcription-translation feedback in circadian rhythm of KaiC phosphorylation. *Science* **307**, 251–254 (2005)
65. O. Wolkenhauer, S.N. Sreenath, P. Wellstead, M. Ullah, K.H. Cho, A systems and signal-oriented approach to intracellular dynamics. *Biochem. Soc. Trans.* **33**, 507–515 (2005)
66. SRSim web site. <http://www.biosys.uni-jena.de/Members/Gerd+Gruenert/SRSim.html>
67. Model repository and download area. <http://www.molecular-computing.de> via menu item Veröffentlichungen/Publications

## Chapter 6

# Biochemical Networks Discrete Modeling Inspired by Membrane Systems

John Jack, Andrei Păun and Mihaela Păun

**Abstract** The ideas expressed in this work pertain to biochemical modeling. We explore our technique, the Nondeterministic Waiting Time algorithm, for modeling molecular signaling cascades. The algorithm is presented with pseudocode along with an explanation of its implementation. We discuss several important extensions including: (i) a heap with special maintenance functions for sorting reaction waiting times, (ii) a nondeterministic component for handling reaction competition, and (iii) a memory enhancement allowing slower reactions to compete with faster reactions. Several example systems are provided for comparisons between modeling with systems of ordinary differential equations, the Gillespie Algorithm, and our Nondeterministic Waiting Time Algorithm. Our algorithm has a unique ability to exhibit behavior similar to the solutions to systems of ordinary differential equations for certain models and parameter choices, but it also has the nondeterministic component which yields results similar stochastic methods (e.g., the Gillespie Algorithm). There are several extensions for the current work discussed at the end of the chapter.

---

J. Jack

Bioinformatics Research Center, NC State University, Raleigh, NC 27695, USA  
e-mail: jrjack@ncsu.edu

A. Păun (✉)

University of Bucharest, Str. Academiei nr.14, sector 1, 010014 Bucharest, C.P., Romania  
e-mail: apaun@fmi.unibuc.ro

M. Păun

Bioinformatics Department, National Institute of Research and Development for Biological Sciences, Splaiul Independenței, Nr. 296, Sector 6, Bucharest, Romania  
e-mail: mpaun@latech.edu



## 6.1 Introduction

The focus of this chapter is the simulation of molecular signaling cascades, specifically the Fas-mediated apoptotic pathway. The discussion centers around living systems—e.g., biological cells—and the changes in their biochemical compositions, which is brought about through the various interactions of the intracellular proteins. Indeed, signal transduction (and molecular signaling cascades) describes the systematic interactions of different cellular proteins, starting with some sort of signal (e.g., an external ligand binding to a cell surface receptor) and reaching some sort of endpoint (e.g., the upregulation of a protein eliciting physical changes to the cell). We explore the molecular mechanisms behind biochemical evolution with implied physiological responses through computational modeling.

There has been significant interest recently for using computer science tools to address and understand the relevance of the genomic biological data. Besides the modeling of molecular signaling cascades discussed herein, there is a large thrust into protein-folding prediction. Computer algorithms have been designed to decipher the three-dimensional folding of a protein based on its amino acid sequence. Understanding this structure will give insight into the function, since these two elements—structure and function—are intrinsically linked. With the sequencing of the entire human genome, scientists are looking to computer models in order to understand all that the Human Genome Project and similar efforts have uncovered for us.

Although the number of genes encoded in a cell is alarmingly small, given the vast physical differences in the life of this planet, we look for another aspect to explain the diversity of living things. To understand ourselves, we need to quantify the interactions of the things encoded by the genes. The complexity of the machinery we call life stems from relatively basic components (proteins) interacting through intricate reaction networks. Simplicity breeds complexity.

For the design of the NWT algorithm, we have chosen a biologically inspired underlying framework: Membrane Systems (or P Systems). The essential design goal was to create a new simulation technique capable of exhibiting qualities comparable to the stochastic methods—e.g., the Gillespie algorithm [22, 23]—but also systems of ordinary differential equations, depending on the particular configurations of the systems being modeled. Moreover, the NWT algorithm is designed to be less computationally intensive than the Gillespie algorithm; however, NWT maintains a level of nondeterminism that allows divergent solutions compared with systems of ordinary differential equations.

Other approaches have been considered in this area of simulating cells using membrane systems, notable is the work of V. Manca from [49].

For the rest of this Sect. 6.1 we will introduce some of the existing techniques for modeling the dynamics of intracellular proteins in a reaction network. Differential equations have been the predominate form of modeling for a very long time. However, a competing algorithm was developed in the late 1970s using stochastic fluctuations (and discrete mathematics rather than continuous mathematics) to more accurately

predict protein dynamics in [22]. Since then, both methods have been improved, developed, modified, adapted and even combined to give us fast, accurate simulations of molecular signaling cascades.

### 6.1.1 Modeling with Differential Equations

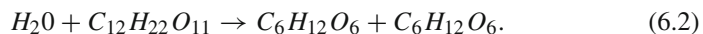
Systems of ordinary differential equations are employed to model a wide range of phenomena, including but not limited to modeling the dynamics of molecules in a biochemical reaction network. In fact, due to their heavy mathematical background and the speed at which they can be solved, systems of ordinary differential equations are quite popular in the modeling of molecular signaling cascades. However, a recurrent theme in this work will be the fact that molecular modeling with ordinary differential equations does not always yield desirable results. Moreover, the solutions to systems of ordinary differential equations can often yield misleading results, failing to accurately represent the minority behavior of a few cells in favor of results illustrating the average behavior of the majority of cells. In later sections we will continue to comment on this shortcoming of the differential equations. For now, we will briefly discuss how we can model biochemical systems with systems of ordinary differential equations.

We set up an ordinary differential equation for each type of molecule in the system. For each species  $X_i$ , we have

$$\frac{dX_i}{dt} = f_i(X_i, \dots, X_n), \quad (6.1)$$

where  $f_i$ 's are functions (possibly nonlinear and nonhomogeneous). For example, the Hill function, which was initially developed to describe the binding of oxygen to hemoglobin [31], is a classic nonlinear function now having widespread use in describing cooperative binding (such as ligands binding to receptors). In fact, many biological phenomena are being modeled with nonlinear functions.

Essentially, to model the dynamics of protein interactions, we will need a differential equation for each protein with the functions defined according to how the proteins react to each-other in the system. For example, we consider an early investigation into chemical equilibrium, which was made by L. Wilhelmy [51, 87]. His studies were focused on the following sucrose reaction:



If we let  $S(t)$  represent the concentration of sucrose, then Wilhelmy was able to show that

$$-\frac{dS}{dt} = kS, \quad (6.3)$$

where  $k$  is a *kinetic rate constant*. He designed this equation to match the empirical evidence that the rate of decrease of sucrose concentration was proportional to the concentration remaining unconverted (the *law of mass action* discussed further in Sect. 6.2.1).

If we let  $S_0$ , represent the initial concentration of sucrose, then we have

$$S(t) = S_0 e^{-kt}. \quad (6.4)$$

Since the time of Wilhelmy, chemical kinetics have received a great deal of attention. To address interesting problems in the biochemistry of life, we need much more complex systems. These systems will involve many reactions with a large degree of interdependence. Indeed, we will see increasingly complex systems of ordinary differential equations in the coming sections of this work. This complexity will be driven by the number of interacting elements and reactions, but not necessarily from complicated nonlinearity such as the Hill Function. With more complex systems, we will need to approximate the solutions to the systems of ordinary differential equations. Once the system of differential equations is written out, mathematically describing the molecular mechanisms, it is time to apply approximation methods.

The most common method for approximating a system of ordinary differential equations is the fourth order Runge-Kutta method. This is the method employed by a large majority of computational biology labs. Often, MATLAB is used, and the ode23 and ode45 solvers are built-in functions utilizing Runge-Kutta to provide an approximate solution to the system of ordinary differential equations. For an example of the MATLAB code we have used to determine solutions to systems of ordinary differential equations, we refer the interested reader to [37] (describes the circadian rhythm model from Sect. 6.3).

We have hinted that differential equations are not the only way to model biochemistry. We will now briefly discuss the stochastic techniques for modeling biochemical reaction networks.

### 6.1.2 Stochastic Methods and the Gillespie Algorithm

There are situations where ordinary differential equations fail to adequately represent cellular populations. The biochemical reason for this usually stems from situations of low molecular multiplicity. In one of the most important works on stochastic approaches to chemical kinetics, McQuarrie [51] provided a rich description of the historical background to stochastic techniques as well as some exactly solvable systems.

Kramers was the first to use stochastic ideas for modeling the kinetics of chemical equations [45, 53]. The idea of stochastic approaches for modeling chemical systems revolves around the *chemical master equation*. This equation describes the probability of every possible state of the cell (with respect to biochemical composition). Instead of a differential equation for each protein, one would essentially have

a differential equation for every possible state of the cell. For very small systems, the chemical master equation can be solved directly (see [51] for some systems). However, it becomes difficult or impossible to directly find the chemical master equation for systems of nontrivial size. It is this reason which led D.T. Gillespie to formulate his now ubiquitous algorithm for exactly solving the chemical master equation.

Gillespie published two landmark papers in 1976 and 1977. In [22], he presents the framework for an exact stochastic method, which accurately predicts the chemical master equation. Then, in [23], Gillespie describes the Stochastic Simulation Algorithm (SSA); the algorithm is now aptly named the Gillespie Algorithm, and it is the most commonly applied/adapted technique for stochastic simulation of biochemical networks. The Gillespie Algorithm is at the heart of most discussions on stochastic modeling.

In [23], Gillespie discusses two important points on the failure of classical modeling (differential equations); the approach assumes that the time evolution of a chemically reactive system is both continuous and deterministic. However, in nature, chemically reacting systems evolve in a discrete manner, since molecular multiplicities can obviously change only by integer amounts.

We will forgo an explanation of the algorithm, since it has been reported in the literature. However, we would like to mention the main limitation of the algorithm. As stated in the original paper [23], the Gillespie Algorithm places a high premium on the speed of the computer's CPU. The limitations are dependent on the number of reactions in the system. Also, the algorithm requires multiple runs to correctly quantify the system. This works in conjunction with the speed limitations, making stochastic simulations an enduring process.

### 6.1.3 Improving the Gillespie Algorithm

Since its creation in 1977, the Gillespie Algorithm has been the focus for improvements in efficiency. The most notable improvement for the Gillespie Algorithm comes from the work of Gibson and Bruck [21]. They were able to reduce the computational complexity of the algorithm considerably, through the addition of a method for sorting the reactions and reducing the dependence on random number generation. However, the limitation associated with reaction network growth—number of molecules and reactions—is still an issue.

A number of methods have now been proposed to combine differential equations with the Gillespie Algorithm. These hybrid methods attempt to divide the reactions into *fast* and *slow*. We will not provide an exhaustive discussion on each method, but we wish to mention two notable works below.

The work of Haseltine and Rawlings [29] has been well-cited. They provide the theoretical background for dividing reactions into fast and slow subsets, allowing for the fast reactions to be approximated either deterministically or as Langevin equations. Essentially, they are able to integrate the system over much larger time steps than the original Gillespie Algorithm. For the original Gillespie Algorithm,

increasing the number of molecules for a fast-reacting protein will significantly increase the computational load; however, by using deterministic processes for fast reactions, the computational load of their algorithm will not increase in this case.

Rao and Arkin [71] applied the quasi-steady state assumption to modify the Gillespie Algorithm. Using the quasi-steady state assumption, they were able to reduce model complexity by reducing the number of molecular species and reactions. Essentially, the assumption is that the net rate of formation is approximately zero for highly reactive and transitory species—e.g., enzyme-substrate complexes. In their paper, the authors provide some mathematical rigor behind the algorithm as well as some results for example systems.

6.1.4 Our Work

In Sect. 6.2, we have provided an introduction to the concept of modeling biochemical reaction networks and a brief discussion on some of the most popular techniques. For the rest of this work, we will focus on biochemical modeling with our algorithm, NWT algorithm that was inspired by the membrane systems area.

In Sect. 6.2.1, we will introduce the NWT algorithm. We provide an in-depth look at the pseudocode; a discussion on the implementation; and some results from example systems illustrating the concepts of the algorithmic improvements are provided in the chapter. Section 6.3 illustrates the results of two popular models: the Lotka-Volterra predator-prey model and a circadian rhythm model. The two models are used to emphasize the difference between our algorithm and the solutions to systems of ordinary differential equations and the Gillespie Algorithm.

6.2 Membrane Systems as Cell Simulators

The NWT algorithm is using the discrete nature of the P systems for simulating the intra-cellular processes. We simulate the biochemistry of a cell as the evolution of the Membrane System. There are a variety of intracellular biochemical reaction types; we provide a few examples in Table 6.1, in order to facilitate the understanding of associating time with Membrane System evolution.

In order to plausibly model the biochemistry of life, individual biochemical interactions need to occur asynchronously over different lengths of time. Our Mem-

Table 6.1 Typical examples of biochemical reactions

$R_1$ : Monomolecular decay:	$A \xrightarrow{k_d} \emptyset$
$R_2$ : Monomolecular reaction:	$A \xrightarrow{k_x} B$
$R_3$ : Bimolecular reaction:	$A + B \xrightarrow{k_y} C$
$R_4$ : Trimolecular reaction:	$A + B + C \xrightarrow{k_z} D$

brane System employs the *law of mass action*. The law states that reaction rate is directly proportional to the number of reactants available in the system. In other words, the time required to execute a rule in the Membrane System is dependent on the number of its reacting species.

The law of mass action gives us the power to temporally describe the evolving configurations of our Membrane System. To understand the asynchrony of rule execution, we need to discuss the *kinetic rates* pertaining to the law of mass action. The kinetics of a chemically reactive system are often described as concentration-based values. This is common for the types of experiments used to derive the rates, typically involving enormous populations (millions) of cells. The cells are often lysed as a large population, molecules are measured in terms of light intensity—e.g., radiological or photonic markers—and data are given as concentrations of species across cell population. These values can be averaged across the cell population, yielding concentrations per cell. We rely on these values to fit our models, but the values are derived from entire cell populations instead of individual cells. Hence, the interesting phenotypic, biochemical and physiological characteristics of individual cells can be sometimes overgeneralized (or lost) in lieu of the behavior of the majority of the cells in the population.

Some labs employ techniques to measure single-cell dynamics. For example, interesting results/models on p53 have been reported [46]. In [46], the authors showed individual cells undergo not dampened oscillations, as reported in [3], but each individual cell instead exhibits a different numbers of oscillations. The average behavior for the cell population appeared to be dampened, but individual cells did not behave this way.

The concentration-based kinetic rates,  $k_R$  for each rule  $R$ , will have units nMs,  $\mu$ Ms, etc. Assuming we have accurate kinetic information for all rules in our Membrane System, we set up multiplicity-based kinetics—a discrete kinetic constant,  $const_R$  for each reaction  $R$ . When we initialize our Membrane System, we calculate the discrete kinetic constants from the concentration-based kinetic rates in the following way:

$$const_R = \frac{k_R}{V^{i-1} \times N_A^{i-1}}, \quad (6.5)$$

where  $R$  is the reaction,  $V$  is the volume of the system,  $N_A$  is Avogadro's constant ( $6.0221415 \times 10^{23}$ ) and  $i$  is the number of reactant species involved.

Using the law of mass action and discrete kinetic constants we can define the *Waiting Time (WT)* of a reaction in the Membrane System. The WT is a value assigned to each reaction, signifying the next timepoint for a single execution of the reaction. As molecular multiplicities will change throughout a simulation, from one configuration to the next, so will the WTs of reactions utilizing those molecules. The equations for calculating the WT of a first order, second order, and third order reaction are provided in Eqs. 6.6, 6.7, and 6.8. The rules are those described in Table 6.1.

$$WT_{R_1} = \frac{1}{k_d * |A|}, \quad (6.6)$$

$$WT_{R_3} = \frac{1}{k_y * |A| * |B|}, \quad (6.7)$$

$$WT_{R_4} = \frac{1}{k_z * |A| * |B| * |C|}, \quad (6.8)$$

where  $A$ ,  $B$ , and  $C$  are the reactants required for reactions  $R_1$ ,  $R_3$  and  $R_4$ ,  $|A|$ ,  $|B|$ , and  $|C|$  represent the number of molecules present in the system at the moment of WT calculation, and  $k_d$ ,  $k_y$ , and  $k_z$  are the discrete kinetic constants.<sup>1</sup>

Using Eqs. 6.6, 6.7 and 6.8 we can calculate the WT for each reaction in the Membrane System. Reactions with more than three reactants can be reduced into multiple sub-reactions of lower order. We used a min-heap for sorting reactions, where the top of the heap is the reaction with the smallest WT—i.e., the next reaction to be executed. However, we need to use nonstandard methods for maintaining the heap, due to the asynchrony of the rules and the sharing of reactants. These non-standard methods are similar to those proposed by Gibson and Bruck [21] in their modification to the Gillespie Algorithm.

To clarify, when a rule is applied, multiple nodes can have changes to their Waiting Time, since the multiplicities of particular species of the system have changed. These species can be shared over multiple reactions. Hence, multiple Waiting Times potentially can fail the min-heap property throughout the tree simultaneously at each new configuration. In order to handle this, we use heap maintenance methods similar to those proposed by Gibson and Bruck [21] in their modification of the Gillespie Algorithm.

### 6.2.1 Description of the NWT Algorithm

The Membrane System is a mathematical description of the entire biochemical model, defining the reactions, cellular compartments, molecular species and multiplicities, etc. We will now provide the steps for the NWT algorithm.

- I *Build Membrane System*: Import information for Membrane System—alphabet, membrane hierarchy, etc. Convert protein concentrations to molecular multiplicities. Convert kinetic rates to discrete kinetic constants. For each reaction  $R_i$ , where  $1 \leq i \leq m$ , we calculate the initial Waiting Time,  $WT_{R_i}$ . Choose the

---

<sup>1</sup>  $R_1$  and  $R_2$  from Table 6.1 are calculated in the same way (replacing  $k_d$  with  $k_x$ ) because they both have reaction order one and use the same reactant species, albeit the products are different. If one of the reactants for a reaction has no molecules present in the system, then we set the WT equal to infinity. We have implemented the algorithm in ANSI C, so we automatically have  $\frac{1.0}{0.0} = \infty$ .

desired amount of time for the simulation,  $\tau_{fin}$ . Set current simulation time to zero ( $\tau = 0$ ).

- II *Build Heap*: Using the reaction Waiting Times, we build a min-heap of all reactions in the system.
- III *Select Rule*: Choose the reaction with the lowest Waiting Time—the top of the min-heap. Upon selecting the root node of the heap, recursively check to see if there are any children nodes sharing the minimum Waiting Time. If such a tie for minimum Waiting Time exists, then proceed to Step IV. If no tie exists, then continue to Step V.
- IV *Handle Tie*: Check the multiplicities of the reactant species for all tied reactions. If there are enough reactants to satisfy all of the reactions with the minimum Waiting Time, implement all tied reactions. If there are not enough reactants to accommodate all the reactions, we make a nondeterministic choice on the rules to apply until no more rules can be applied.
- V *Apply Rule*: Update the multiplicities of the reactant(s) and product(s) for the reaction(s) from Step III. Aggregate the simulation time ( $\tau = \tau + WT_{applied}$ ).
- VI *Update Rules*: Recalculate the Waiting Time for all reactions whose reactant(s) include the product(s) or reactant(s) of the applied reaction(s). That is, we need to see how the multiplicity changes from the applied reaction(s) have affected the Waiting Times for each rule dependent on those proteins with a new multiplicity. For each such reaction compare the new Waiting Time with the existing Waiting Time and keep the smallest of the two<sup>2</sup> (unless the new time is infinity, indicating insufficient reactants for a reaction to occur).
- VII *Memory Enhancement*: If the recalculation of a reaction's Waiting Time results in a value of infinity, then we must store the amount of time waited as a percentage ( $Mem_{perc}$ ). If the recalculation of a reaction's Waiting Time results in a real value and the previous value was infinite, then the Waiting Time will need to be adjusted according to the stored memory percentage.
- VIII *Heap Maintenance*: Adjust the min-heap, bubbling reaction nodes up or down in order to satisfy the min-heap property, once reaction Waiting Times have been recalculated according to the multiplicity changes.<sup>3</sup>
- IX *Termination*: If  $\tau = \tau_{fin}$ , then terminate the simulation. Output the multiplicity information for entire simulation. Otherwise, go back to Step III.

Our simulation technique is modular; we initialize the NWT with models encoded in the Systems Biology Markup Language (SBML). SBML is one of the most popular methods to encode biochemical models. It is developed through a broad international collaborative effort with many cooperating institutions [33]. To set up the SBML files, we use the CellDesigner software [19, 20], which is maintained through Keio University.

<sup>2</sup> If at this moment the waiting time is increasing—reactants of this rule have been used—we keep the old waiting time considering that the proteins have already started binding, and none of them was used in the other reaction.

<sup>3</sup> To accommodate the multiple changes in Waiting Times, we employ nonstandard heap maintenance methods.



With the SBML code, and the kinetic constants/WT calculations described earlier, we have the information for the initialization of the system. This sufficiently describes Step I of the NWT algorithm. In ANSI C, we can store all of the information describing the Membrane System in two arrays of structs: an alphabet array and a rules array. For Step II, we build a min-heap in the standard bottom-up way. For any two reactions,  $R_1$  and  $R_2$ , in the heap, if  $R_2$  is a child of  $R_1$ , we have

$$WT_{R_1} \leq WT_{R_2}. \quad (6.9)$$

Step III of the NWT algorithm, selecting a rule, requires  $O(1)$  time to complete, since the WTs were sorted in Step II. However, we must check to see if any other rules have the same WT. If two or more rules are attempting to execute at the same instant of time, we must ensure there are enough reactants to execute all competing rules. If insufficient numbers of molecules exist for all the potentially executable rules, then we must nondeterministically apply reactions until all available reactants have been exhausted. We create an array pointing to all rules slotted to occur at the next moment in time. If there is only one element in the ties array—no competition for resources—we can move on to Step V, skipping Step IV, and execute one rule. Otherwise, we proceed with Step IV.

To nondeterministically apply rules, we randomly generate numbers between 0 and the end of the ties array. Using this randomly chosen index, we check if sufficient reactants exist to implement the reaction. If there are sufficient reactants, we apply the reaction—i.e., we increase the multiplicity of the product(s) by one and decrease the multiplicity of the reactant(s) by one. If there are insufficient reactants, we skip the reaction, and no multiplicity changes occur for the reaction. In either case, the reaction is removed from the ties array, and the process continues until the ties array is empty. This completes the discussion for Steps IV and V of the algorithm. Remember, in the case of only one reaction, we skip Step IV and apply just the one reaction in Step V. Either way, we are ready to move on to Step VI: Update Rules.

For each reaction applied in Step V, we must recalculate the WT of the applied reaction and the WT of every reaction affected by the multiplicity changes. We must discuss Step VI within the context of the heap maintenance. Hence, we will continue the discussion of Step VI in Sect. 6.2.2, which will continue with the discussion of Step VIII, Heap Maintenance. As we will see in the next two sections, Steps VI, VII, and VIII are intertwined. The discussion of Step VII is found in Sect. 6.2.3.

## 6.2.2 Maintaining the Min-heap

As previously discussed, heap maintenance is completed in a nonstandard way. In a typical min-heap implementation, the top node is selected and removed from the heap. Meanwhile, new nodes are added to the bottom, bubbling the nodes up in order to satisfy the min-heap property. There are two reasons why we do not wish to remove the applied rules from the top of the heap and add them to the bottom.

First, the number of reactions will not grow or shrink during a simulation run. Therefore, it is unnecessary to remove/add reactions to the heap after initialization. Second, a reaction, once applied, typically has a new WT very similar to the previous value. The new WT will often need to be located near the top of the heap, once the heap is adjusted to satisfy the min-heap property. Removing the top node, adding it to the bottom, and bubbling it back to near the top would be a waste of clock cycles, especially for a significantly large numbers of reactions.

After recalculating the WT for each affected rule—a reaction requiring reactants with changed multiplicity due to the execution of a rule—we reposition the node in the heap. The algorithm handles repositions nodes one at a time until all necessary nodes have new WTs and correct positions in the min-heap.

With the heap implementation, we were able to effectively reduce the computational complexity of our previous technique [9] with respect to the asymptotic complexity from  $O(n^2)$  to  $O(n * \log(n))$ . There are several assumptions we can make, to define our computational complexity, which are typical for many signaling cascades:

1. Each reaction involves a maximum of five proteins;
2. The number of reactions having the same reactant is bounded (usually 3, at most 5);
3. Due to the nature of chemical kinetics, the number of tied reactions is very small.

With 1, 2 and 3 we can state our complexity as  $O(n \log n)$  (with respect to the number of reactions simulated).

Technically, Steps VI, VII and VIII occur at the same time. In this section, we have described Steps VI and VIII. Step VII merely factors into the calculations of the WTs for affected rules. In order to understand the memory enhancement of the NWT algorithm, we will consider a small example system.

6.2.3 Memory Enhancement

There are often situations in biochemical networks, where one (or more) protein(s) is a reactant for two or more reactions of different kinetic rates (fast vs. slow). In order to explain our *memory enhancement*, we will consider an example system involving only three reactions ( $R_1$ ,  $R_2$  and  $R_3$ ) acting on four proteins ( $A$ ,  $B$ ,  $C$ , and  $D$ ). The model is described in Table 6.2.

Table 6.2 An example system to illustrate memory enhancement

Reaction	Rate Constant	Initial Molecules
$R_1: A \rightarrow C$	$k_1$ (slow)	$A = 1$
$R_2: A \rightarrow B$	$k_2$ (fast)	$B = 0$
$R_3: D \rightarrow D + A$	$k_3$	$C = 0$
		$D = 1$

When described as a system of ordinary differential equations, the biochemical network is given in Eq. 6.10.

$$\begin{aligned}
 \frac{d[A]}{dt} &= -k_1[A] - k_2[A] + k_3[D] \\
 \frac{d[B]}{dt} &= k_2 * [A] \\
 \frac{d[C]}{dt} &= k_1 * [A] \\
 \frac{d[D]}{dt} &= 0
 \end{aligned} \tag{6.10}$$

We specifically designed the system above to highlight the effects of the memory enhancement, and we will compare our refined algorithm with solutions to the system of ordinary differential equations described in Eq. 6.10. A model similar to this one could be used to investigate the dynamics of human immunodeficiency type 1 (HIV-1) Tat protein, since it is initially transcribed at very low numbers [40]. Once Tat is assembled in the cytosol, it can be exocytised or translocated to the nucleus [76]. When Tat is translocated to the nucleus it can begin upregulating HIV-1 proteins (including itself). Since the downstream effects of Tat translocation to the nucleus has profound impacts on the cell (causing upregulation of the HIV-1 proteins), a discrete and nondeterministic approach is necessary to follow the dynamics of the low levels of Tat proteins [86].

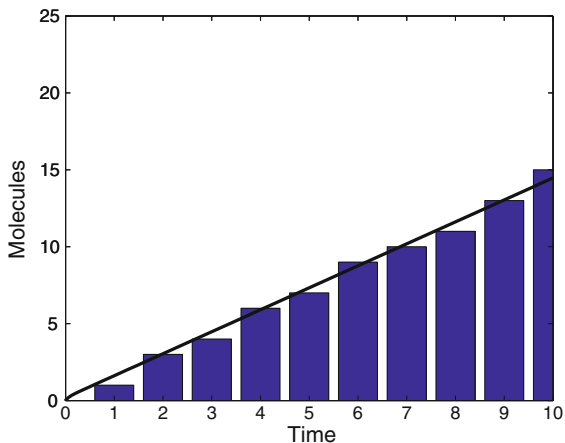
In the system, molecules of  $A$  are formed from molecules of  $D$ . This reaction can basically be viewed as a combined transcription and translation rule with  $D$  being the gene and  $A$  being the protein encoded by the gene. Once a molecule of  $A$  is formed, it has the option doing one of two things: (i) turning into a molecule of  $B$  at rate  $k_2$  or (ii) turning into a molecule of  $C$  at a rate  $k_1$ . If we consider the species  $A$  as being analogous to HIV-1 Tat protein, then  $A \rightarrow B$  could be translocation to the nucleus and  $A \rightarrow C$  could be translocation to the extracellular environment.

Next, we will look at two cases for the model described in Table 6.2 and discuss the memory enhancement. The cases vary by choices of the kinetic rates. The first case shows that the memory enhancement can produce the same results as the deterministic differential equations approach. For the second case, we will show how the technique can produce different results, illustrating that the ability of the NWT algorithm to explore nondeterminism of molecular signaling cascades. There are no modifications to the NWT algorithm between the two cases. The only difference is the kinetic constants for initialization of the model.

### 6.2.4 Case 1: Deterministic Memory Enhancement

For the first case, we let  $k_1 = 10$ ,  $k_2 = 4$ , and  $k_3 = 5$ . The results of a simulation using the NWT algorithm plotted against the solution of the system of ordinary differential equations is shown in Fig. 6.1. The graph shows the accumulation of  $C$

**Fig. 6.1** The number of  $C$  molecules for a simulation of  $t = 10$ . The NWT algorithm results are represented with bars while the solution to the system of ordinary differential equations is the *solid line*



molecules throughout a 10 s run. The bars of the graph are the discrete simulation results using the NWT algorithm, and the black line is the solution to the system of ordinary differential equations. With the choices of the kinetic values, there are no nondeterministic decisions for the entire length of the NWT simulation. So, there is no surprise that the NWT algorithm shows the same increase in  $C$  molecules as the solution to the system of ordinary differential equations.

At initialization ( $t = 0$ ), there is exactly one molecule of  $D$  and one molecule of  $A$ . Therefore, from Eq. 6.6, we see that all three reactions have real (finite) waiting times when the simulation begins. Furthermore, we have  $WT_{R_1} = 0.25$ ,  $WT_{R_2} = 0.1$  and  $WT_{R_3} = 0.2$ . Using these  $WT$ s, we are able to build a min-heap where the top node is  $R_2$ , since it has the smallest Waiting Time.

The first reaction to occur is  $R_2$ , which immediately exhausts the system's supply of  $A$  molecules, yields one molecule of  $B$  and a simulation time of  $t = 0.1$ . The rules affected by the applied rule must be recalculated; the Waiting Times are now  $WT_{R_1} = WT_{R_2} = \infty$ . Since  $R_3$  does not require a molecule of  $A$ ,  $WT_{R_3}$  is left unchanged after the first reaction is executed. Also, upon readjusting the min-heap,  $R_3$  is now at the top since it has the smallest value for  $WT$ .

The next reaction to be applied is  $R_3$ , which gives us a new molecule of  $A$  and a simulation time of  $t = 0.2$ . This is where the memory enhancement plays a role. In the first step, reaction  $R_2$  used up all of the molecules of  $A$ . However, when this happened,  $R_1$  had already *waited* for 0.1 s—the amount of time a molecule of  $A$  was in the system. The memory enhancement allows the simulator to keep track of the percentage of time waited. In other words,  $R_1$  waited for 0.1 s out of its required 0.25 s, which means it has waited 40 % of its Waiting Time. If we allow the algorithm to keep track of this percentage, then, when a new molecule of  $A$  is formed, we can recalculate the  $WT$  for reaction  $R_1$ , using the percentage to adjust its Waiting Time accordingly. That is, after  $R_3$  is applied in step two and we have a new molecule of  $A$ , we recalculate  $WT_{R_1}$  using Eq. 6.6, but we take 60 % of this number—the percentage of time left to wait.

To recap, the first reaction applied is  $R_2$ . The Waiting Times for reaction  $R_2$  and  $R_1$  are both recalculated as infinity, since  $R_2$  used all the molecules of  $A$  present in the system. The memory enhancement allows  $R_1$  to store the percentage of time it has left to wait (60%) when  $A$  is exhausted. Next,  $R_3$  is the second reaction to be applied. When this occurs, there is a new molecule of  $A$  in the system, which means  $R_2$  and  $R_1$  need to be recalculated. The Waiting Time of  $R_2$  is calculated as 0.1, but the Waiting Time of  $R_1$  is recalculated as 0.15. This number stems from the equation

$$WT_{R_3} = Mem \frac{1}{k_3 * |A|} \quad (6.11)$$

where  $Mem$  is the percentage of time left to wait (60% in the example above). We will reiterate the memory enhancement calculations in the second case study with different kinetics.

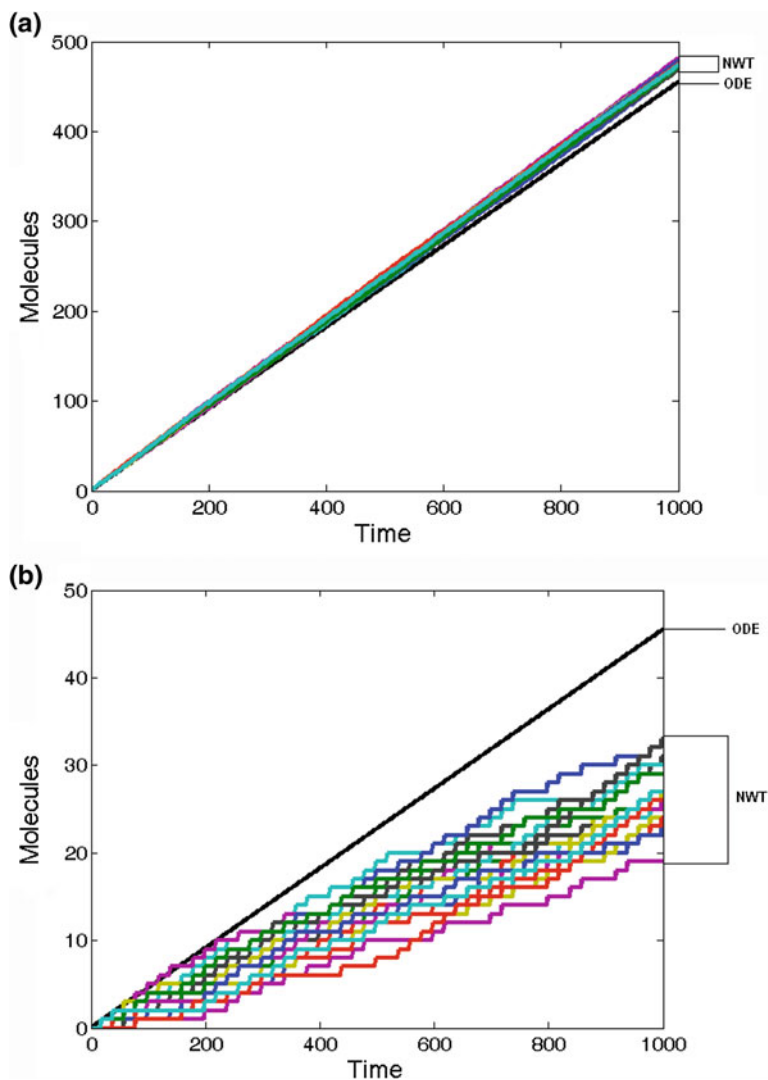
In this case, the solutions to the system of ordinary differential equations and the results from the NWT algorithm agree. In the next case, we will explain how the nondeterminism of the NWT algorithm can lead to results different than continuous, deterministic solutions to system of ordinary differential equations.

### 6.2.5 Case 2: Nondeterministic Memory Enhancement

We will now modify the kinetic constants to highlight the effects of the nondeterministic component of the NWT algorithm in conjunction with the memory enhancement. Although the kinetics of our sample system have been deliberately chosen to highlight the nondeterministic effects, we will later show, in Sect. 6.3, how our nondeterministic logic can have similar implications in a known model, comparable to the Gillespie Algorithm—deviating from deterministic simulations—but at a considerably reduced computational cost.

We need to note that the NWT algorithm will run faster than the Gillespie Algorithm, but will lose from the sensitivity of the simulation with respect to Gillespie since it will deviate from the chemical master equation. Its usefulness will be in being able to simulate larger pathways and even several pathways at the same time integrating much more reactions in the system. In such a case Gillespie's algorithm would be unable to finalize simulations in acceptable time, while the NWT algorithm will be able to provide useful solutions based on discrete mathematics (not continuous mathematics).

For our next simulations, we assume  $k_1 = 0.1$ ,  $k_2 = 1.0$ , and  $k_3 = 0.5$ . The initial Waiting Times are initialized as  $WT_{R_1} = 10$ ,  $WT_{R_2} = 1$ , and  $WT_{R_3} = 2$ . In Fig. 6.2, we see the accumulation of  $B$  and  $C$  molecules. The results of the ODE-based simulation are visibly different than the results of the NWT algorithm. The reasons for the differences are the nondeterministic decisions on reaction competition for  $A$  molecules.



**Fig. 6.2** Results of the memory enhancement simulation for the ordinary differential equations and the NWT algorithm. **a** Number of molecules of species *B* and **b** number of molecules of species *C*

In both graphs we see the results of the solution to the system of ordinary differential equations (straight black line) shown with many runs with the NWT algorithm. In the two graphs, we have (a) the number of molecules of *B* and (b) the number of molecules of *C* are shown. Molecules of *B* and *C* both come from *A* molecules. However, the reaction for *B* is faster than the reaction for *C*. In the solution to the system

of ordinary differential equations, a molecule of *A* can be used to partially satisfy *B* and *C*. Since our NWT algorithm is discrete, the molecules are nondeterministically chosen to satisfy one or the other—similar to cellular processes. The reaction changing *A* into *C* ‘remembers’ how long it has waited, and uses this information the next time a molecule of *A* is ready.

Based on the initialized Waiting Times, the first reaction to be applied is  $R_2$ . After  $R_2$  is applied, the simulation time is aggregated ( $t = 1$ ) and there are no more molecules of *A* present in the system. Similar to the previous case study, the Waiting Times for  $R_2$  and  $R_1$  are both set to infinity (no reactant molecules available). However, we store the percentage of time waited ( $Mem$ ) for the slow reaction  $R_1$ —in this case,  $Mem_{R_1} = 90\%$ . Since  $R_3$  is unaffected by the execution of the first reaction, it now has the minimum waiting time. The next rule to be applied is  $R_3$ . The simulation time is adjusted ( $t = 2$ ), and we now have a new molecule of *A*. With our new *A* molecule available, we must recalculate the Waiting Times for  $R_1$  and  $R_2$ .

Using the  $Mem_{R_1}$ , we can recalculate the Waiting Time for reaction  $R_1$  and use the fact that it has already waited 10% of its Waiting Time. Therefore, when a new molecule of *A* is formed 2 s into the run, we recalculate the  $WT_{R_1}$  using Eq. 6.11. In our case, we have  $WT_{R_1} = 9$  and  $WT_{R_2} = 1$ .

In a deterministic sense, our algorithm is capable of generating equivalent results to an ordinary differential equations model (see case 1 above). But, with the nondeterminism of our algorithm, the memory enhancement can lead to different results. Continuing the calculations for the simulation, we skip ahead to a future event ( $t = 18$ ). Up until this point, we have been creating molecules of *A*, and every single one of them has been deterministically chosen to change into molecule *B* via reaction  $R_2$ . But, at  $t = 18$ , a molecule of *A* has been created, and the Waiting Times of reaction  $R_1$  and  $R_2$  are equal  $WT_{R_1} = WT_{R_2} = 1$ . The reason for this is that we have  $Mem_{R_1} = 10\%$ . In other words,  $R_1$  and  $R_2$  are competing to use the same single molecule of *A* to form a *C* and *B* molecule, resp. The solution to the system of ordinary differential equations has no issue at this timepoint, because, whereas our simulator represents molecules of *A* discretely and has only allowed reaction  $R_2$  to occur so far, the differential equations simulation is sending a fraction of each *A* to form a fraction of *B* and *C*. This is merely a consequence to the way solutions to systems ordinary systems of differential equations behave.

Our algorithm faces the question: at  $t = 18$  should the *A* molecule be allowed to satisfy  $R_1$  or  $R_2$ ? The algorithm answers the question by making a nondeterministic choice between  $R_1$  and  $R_2$  (step 3 of the NWT algorithm). If  $R_1$  is chosen, then it is applied, and our results stay with the ordinary differential equations results (up to  $t = 19$ ). Remember, the ordinary differential equations have been slowly and continuously aggregating the *C* molecules throughout to reach one full molecule of *C* at time  $t = 19$ . However, if  $R_2$  is chosen, then our solution diverges from the previous solution. When the effects of the nondeterministic decisions are aggregated over 1,000 s (15 runs are shown in the figure), we see the different results obtained from the NWT algorithm (Fig. 6.2).

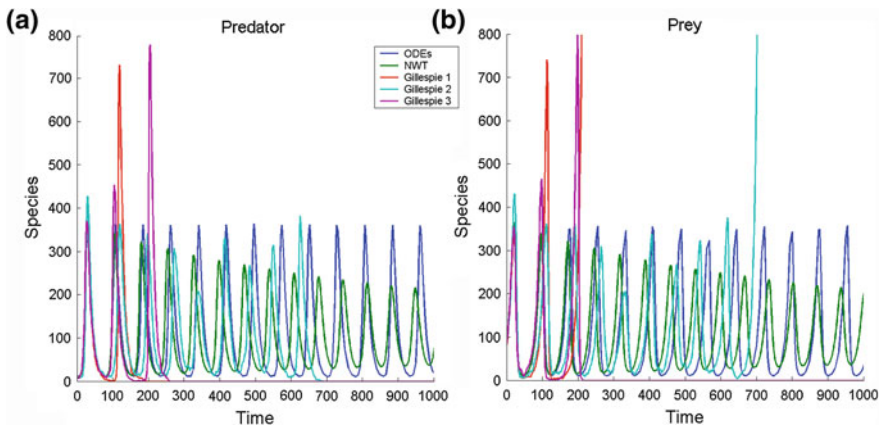
With this example system, we show how reaction memory can affect results. The memory enhancement is designed to give the NWT algorithm results more like solutions to ordinary differential equations in a strictly deterministic sense. However, as a consequence to the reaction memory, we have possible divergence in situations of very low molecular multiplicity, due to the nondeterministic component of the NWT algorithm.

### 6.3 Comparing the NWT Algorithm with the ODE and Gillespie's Algorithm

To emphasize the differences between the NWT, the Gillespie Algorithm, and systems of ordinary differential equations, we will consider two distinct models. First, we consider the classic Lotka-Volterra (predator-prey) model. Described as a system of ordinary differential equations, we have

$$\begin{aligned}\frac{dP_1}{dt} &= P_1 * (a - b * P_2) \\ \frac{dP_2}{dt} &= -P_2 * (c - d * P_1)\end{aligned}\tag{6.12}$$

There are two interacting species—a predator population,  $P_2$ , and a prey population  $P_1$ . Prey species are born at a rate,  $a$ , and consumed at a rate,  $b$ , while predator species are born at a rate,  $d$ , and die at a rate  $c$ . We provide the results for the three simulation approaches in Fig. 6.3. There are two graphs, predator and prey, for the



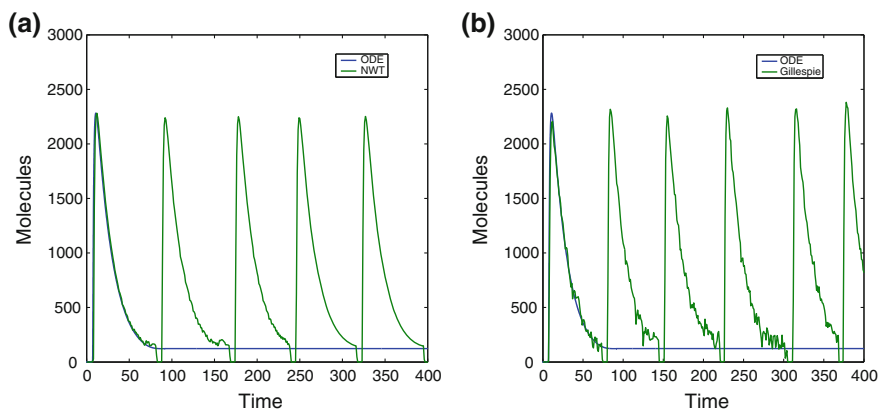
**Fig. 6.3** Results from Lotka-Volterra model depicting dynamics of **a** predator species and **b** prey species



results of the NWT, the Gillespie Algorithm, and the system of ordinary differential equations.

Due to stochasticity, the Gillespie Algorithm will eventually reach zero over a sufficiently long timeline. For instance, runs 1, 2 and 3 show predator reaching zero at  $t = \sim 195$ ,  $t = \sim 250$ , and  $t = \sim 690$  (see Fig. 6.3a). The NWT, on the other hand, generates results comparable to the system of ordinary differential equations—i.e., indefinite oscillations between predator and prey. Although the oscillations for the NWT algorithm appear to be dampened, they eventually settle on a steady oscillating pattern. The difference between the differential equations and the NWT stems from molecular integrity. The NWT algorithm only allows predator or prey species to increase/decrease by integers. Differential equations are continuous, relying on concentrations instead of multiplicities, which is the reason for the differences in the amplitude of stable oscillations.

With this first example system, the NWT algorithm mimics the behavior of the system of ordinary differential equations. We will next illustrate the nondeterminism of the NWT algorithm, deviating from ordinary differential equations. For this system, we chose a popular circadian rhythm model described in [83]. The system of ODEs is provided in Eq. 6.13, and results from all three techniques are shown in Fig. 6.4.



**Fig. 6.4** Results from circadian rhythm model comparing the two nondeterministic approaches **a** The NWT algorithm and **b** the Gillespie Algorithm—with the system of ordinary differential equations

$$\begin{aligned}
\frac{dD_A}{dt} &= \theta_A * D'_A - \gamma_A * D_A * A \\
\frac{dD_R}{dt} &= \theta_R * D'_R - \gamma_R * D_R * A \\
\frac{dD'_A}{dt} &= \gamma_A * D'_R * A - \theta_A * D'_A \\
\frac{dD'_R}{dt} &= \gamma_R * D_R * A - \theta_R * D'_R \\
\frac{dD_{M_A}}{dt} &= \alpha'_A * D'_A + \alpha_A * D_A - \delta_{M_A} * M_A \\
\frac{dA}{dt} &= \beta_A * M_A + \theta_A * D'_A + \theta_R * D'_R - A * (\gamma_A * D_A + \gamma_R * D_R + \gamma_C * R + \delta_A) \\
\frac{dM_R}{dt} &= \alpha'_R * D'_R + \alpha_R * D_R - \delta_{M_R} * M_R \\
\frac{dR}{dt} &= \beta_R * M_R - \gamma_C * A * R + \delta_A * C - \delta_R * R \\
\frac{dC}{dt} &= \gamma_C * A * R - \delta_A * C
\end{aligned} \tag{6.13}$$

where  $A$  and  $R$  represent the number of activator and repressor proteins,  $D'_A$  and  $D_A$  represent the number of activator genes with or without binding to  $A$ ,  $D'_R$  and  $D_R$  represent the number of repressor genes with or without binding to  $R$ ,  $M_A$  and  $M_R$  represent mRNA molecules of  $A$  and  $R$ , and  $C$  represents the corresponding inactivated complex formed by  $A$  and  $R$ .

In Fig. 6.4, we see the similarities between the Gillespie Algorithm and the NWT technique, exhibiting sustained oscillations for the simulation time. The differential equations, since they are strictly deterministic, result in one peak followed by a steady state. With considerably less nondeterministic decisions, the NWT algorithm is able to produce oscillations. To illustrate this fact, we ran three simulations for a longer time span ( $t = 2,000$ ).<sup>4</sup> The number of nondeterministic steps required are given in Table 6.3.

The Gillespie Algorithm requires stochastic decisions at each time step. In contrast, the NWT Algorithm makes nondeterministic decisions only when reactions compete over limited reactants. That is, two or more reactions have the same (minimum) WT, but there are not enough molecules in the system to satisfy all of the potential rules. Less than 0.15 % of the total number of applied rules were

**Table 6.3** Total number of nondeterministic decisions for time = 2,000

Run	Gillespie	NWT
1	3071774	5093
2	3029754	5185
3	3103434	5435

<sup>4</sup> The reason that only three simulations are provided is the excessive time needed for each Gillespie run which limited the number of experiments we could perform.

nondeterministically chosen across all three runs of the NWT algorithm. However, 100 % of the applied rules for Gillespie Algorithm are nondeterministic (for any number of runs). Interestingly, in spite of the relatively small number of nondeterministic decisions, the NWT algorithm is able to exhibit sufficient biochemical noise to induce oscillations similar to the Gillespie Algorithm.

## 6.4 Modeling FAS-Induced Apoptosis

The term apoptosis was coined in the classic work of Kerr, Wyllie and Currie [41]. Within that paper, the authors described apoptosis as a form of cell death distinct from necrosis, citing a lack of inflammation of the tissue among other differences. Apoptosis is often used synonymously with *programmed cell death*, emphasizing a cell's inherent genetic predisposition for death. In other words, the cell's genome contains the instructions for the cell's self-destruction. However, we note that there are other accepted forms of programmed cell death described in the literature [14, 18, 79]. These types of cell death—for example, *aponecrosis*—are characterized as sharing some of the characteristics of necrosis and/or apoptosis. Essentially, apoptosis is a clean and effective method for the elimination of unwanted or damaged cells within the organism. It is sometimes called cellular suicide; the cell receives a message to die and, based on its biochemical composition at the time the death message is received, the cell 'decides' whether to live or die. Furthermore, there are additional situations where apoptosis is not a programmed cell death, such as the case with some cancers and other disorders [27, 50, 56, 58].

Thus, cellular apoptosis is an important process in biological systems. Throughout the entire lifespan of an organism, death by apoptosis is essential for maintaining cellular homeostasis. Indeed, in [72], the authors estimate that a typical human being will produce ten billion cells daily from stem cells to replace the ones dying from apoptosis. There are some especially vital roles for apoptosis in early embryonic development. For example, a developing human initially overproduces the cells of the nervous and immune systems; however, those cells lacking synaptic connections (neurons) or functional antibodies (B cells and T cells) are subject to death via apoptosis (reviewed in [58] and [63], respectively). Aside from a critical early developmental role, apoptosis has also been related to aging effects. For instance, one theory on aging, involving the oxidative stresses on the mitochondria induced by harmful free radicals, illustrates age-related apoptotic cell death [28, 65].

The idea that apoptosis is "programmed" into a cell yields the possibility, by analogy, that we may have the means to reprogram a cell to live or die as needed. In other words, if a cell has cancer, then we can reprogram it to die. If a cell has a virus—e.g., the human immunodeficiency virus or the human papillomavirus—then we can reprogram it to die. In contrast, we may reprogram cells in very close proximity to HIV-infected cells—the so-called bystander cells—to live. These examples may be over-simplified, but they illustrate the importance of understanding genetic manipulation on an *in silico* effort. If we change one gene, we need to be able to predict the consequences of that change. That is at the heart of computational biology.

However, we are not ready to begin manipulating genes involved in apoptosis, because the signaling cascades are too complex. There are too many interacting elements with multiple responsibilities. The only way to gain a deep understanding of the molecular interactions involved in apoptotic signaling is through the development of extensive computer models.

### 6.4.1 *Apoptotic Signaling Cascades*

The study of apoptotic signaling cascades is especially interesting to the fields of biology and medicine since defects in these pathways have been linked to various autoimmune disorders [50], neurological disorders [58], and cancers [27, 56]. Indeed, a recent study [90] implicates Fas-mediated apoptosis in patients with spinal cord injury. Therefore, understanding the molecular mechanisms underlying the apoptosis pathways can offer new therapeutic approaches to combating a wide range of diseases and disorders.

There are two signaling pathways for apoptosis described in the literature: the extrinsic and the intrinsic (or mitochondrial) pathway. The physiological response induced by each pathway is the same—DNA fragmentation, degradation of cytoskeletal and nuclear proteins, formation of apoptotic bodies, etc. However, albeit the two pathways have distinct beginnings, the molecular mechanisms in the final steps of the signaling cascades are the same. The two pathways converge with particular members of a family of cysteinyl-aspartate-specific **proteases**, caspases, which are produced as zymogens—i.e., they require a biochemical change to become active [48].

Depending on the biochemical composition of the cell, these death-inducing stimuli can lead to a proteolytic cascade, whereby the inactive proenzyme caspase are activated and cell death can occur. For instance, Cytotoxic T lymphocytes can send death-inducing ligands (specifically, the Fas ligand) to cells as a method for fighting disease or viral infection. This is the main method through which the body fights disease.

For the rest of this section, we concern ourselves with the molecular mechanisms underlying the Fas-mediated apoptotic signaling cascade—extrinsic and intrinsic pathways.

### 6.4.2 *Fas-Mediated Apoptosis*

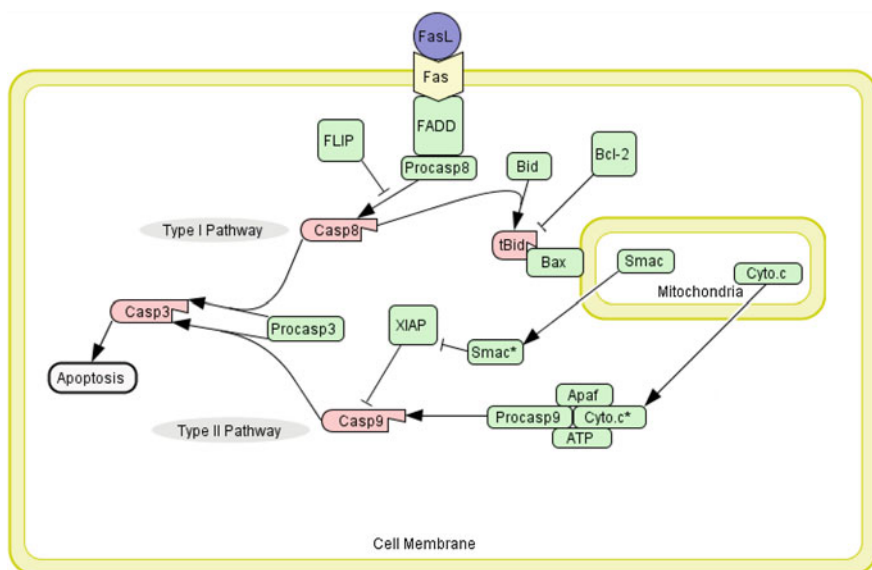
In the past decade, there has been a wealth of information discovered on the Fas-mediated apoptotic pathway. For instance, in [39] they were able to show that Fas/FasL interactions are required for apoptosis of activated T-cells.

One of the troublesome characteristics of some cancerous cells is the upregulation of Fas ligand. This so-called counterattack, is a method the tumor cells can use to delete (by apoptosis) antitumor lymphocytes [59]. There are a variety

of tumor types—e.g., colon cancer, esophageal cancer, melanoma, astrocytoma—showing high expression levels of Fas ligand [60].

In order to test the effectiveness of our technique, we decided to simulate the Fas-mediated signaling cascade, as it was reported in [32]. A graphical representation of the model can be found in Fig. 6.5. The rules are found in [34]. Next, we will walk through the Fas-induced apoptotic signaling cascade. The Fas pathway is most accurately described as two different pathways [75]—type I and type II—sharing an initial phase and an ending phase but unique in the molecular mechanisms in between.

Both the type I and type II pathways begin the same way: the Fas ligand (FasL) binds to a transmembrane receptor, Fas (CD95/APO-1). This receptor is a member of the tumor necrosis factor-receptor super family—a family consisting of over 30 proteins interacting with 19 different ligands. It is expressed on a variety of cells including activated T and B cells. The binding of ligand to receptor is known as receptor cross-linking. When this cross-linking occurs, a conformational change takes place in the receptor producing the complex Fasc. The crosslinking between ligand and receptor, along with recruitment of Fas-associated death domain, form the components of the Death-Inducing Signaling Complex (DISC) [44]. The cytoplasmic domain of this complex recruits Fas-associated death domain (FADD), with a maximum of three FADD per binding site (trimer). While FADD is bound to the complex Fasc, Caspase-8 and FLIP are recruited competitively. Once at least two molecules of Caspase-8 have been recruited to a binding site, a dimer,  $\text{Caspase-8}_2^{P41}$ , is released into



**Fig. 6.5** Picture of the Fas-mediated apoptotic signaling cascade. Both the type I and type II pathways are illustrated

the cytoplasm, where it can then be phosphorylated into active form (Caspase-8\*). The binding of FLIP to the Fasc complex is considered to be an inhibitor of apoptosis, because it decreases the number of sites available for Caspase-8 recruitment.

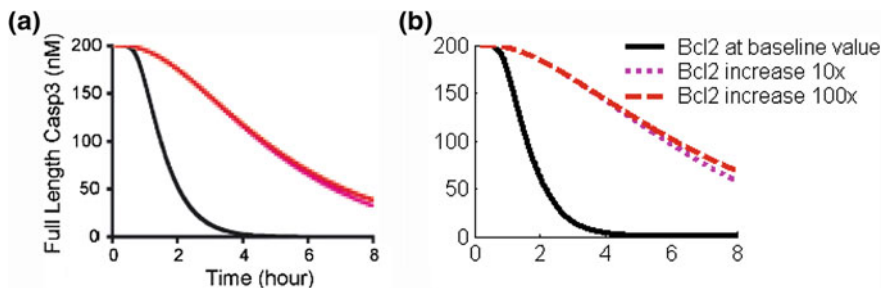
Unless sufficiently inhibited, the signaling cascade can continue in two different ways—the type I or type II pathway. If the initial concentration of Caspase-8 is large enough, Caspase-3 will be directly phosphorylated by the Caspase-8\* (the type I pathway). Otherwise, Caspase-8\* can truncate molecules of Bid (tBid). Each tBid molecule binds with two Bax molecules, which leads to the release of Cytochrome c from the mitochondria (type II pathway). Once released, Cytochrome c binds to Apaf and ATP, forming a complex that can recruit and phosphorylate Caspase-9 (Caspase-9\*). The active Caspase-9\* molecules can continue the cascade by direct phosphorylation of Caspase-3. We consider the activation of Caspase-3 to be the end of the signaling cascade. Hence, from our perspective, the cell is dead once all of the Caspase-3 molecules are activated.

Besides FLIP, there are other inhibiting factors at play: Bcl-2 hinders the release of Cytochrome c from the mitochondria and XIAP blocks Caspase-9\* from binding with Caspase-3. In other words, if sufficient levels of FLIP, Bcl-2, and/or XIAP exist, the apoptotic pathway can be blocked, and the cell lives.

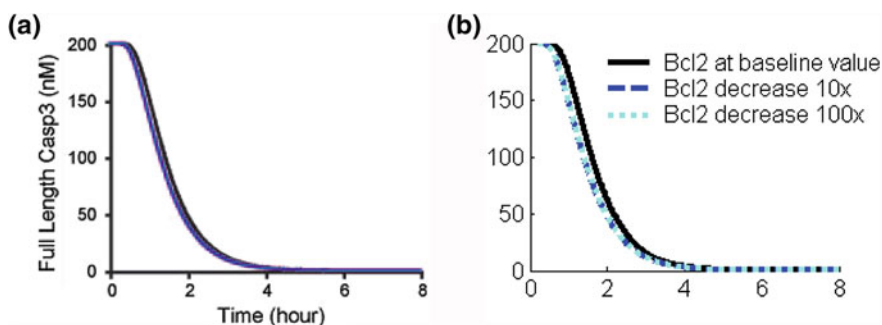
### 6.4.3 Results of Discrete Method

We modeled the pathway described above using 101 different rules working on 53 distinct proteins and protein complexes. Fei Hua et al., in [32], provided the results for the system of ordinary differential equations, as well as some experimental data (from the Jurkat cell line) which they used to fit their model. We compared our results with the results from [32], simulating the same 101 rules and same initial conditions as the system of ordinary differential equations.

Similar to [32], we simulated three different initial concentrations for Bcl-2: the baseline value (75 nMs), an increase by 10-fold (750 nMs), and an increase by 100-fold (7,500 nMs). Assuming a cell volume of  $10^{-12}$  l we converted the concentrations into molecular multiplicities: baseline value (45,166 molecules), 10-fold (451,660 molecules), and 100-fold (4,516,606 molecules). We expected to see a decline in Caspase-3 activation as Bcl-2 concentration was increased by 10-fold and 100-fold; we provide the results of our simulations in (Fig. 6.6). We also provide the results of simulations with a decrease of 10-fold and 100-fold in comparison to the baseline Bcl-2 multiplicity (Fig. 6.7). Notice, the graph based on the NWT algorithm is comparable to the ODE-based results from [32].



**Fig. 6.6** The decline of full length Caspase-3 for different concentrations of Bcl-2: (i) baseline, (ii) 10-fold increase, and (iii) 100-fold increase. **a** The results of the solution to the system of ordinary differential equations and **b** the results of the NWT algorithm

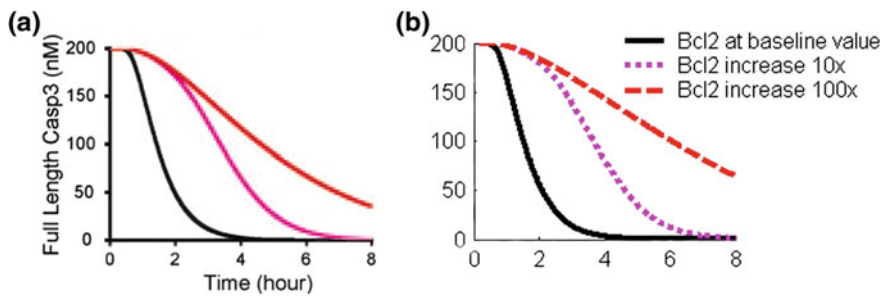


**Fig. 6.7** The decline of full length Caspase-3 for decreased concentrations of Bcl-2: (i) 10-fold and (iii) 100-fold. **a** The results of the solution to the system of ordinary differential equations and **b** the results of the NWT algorithm

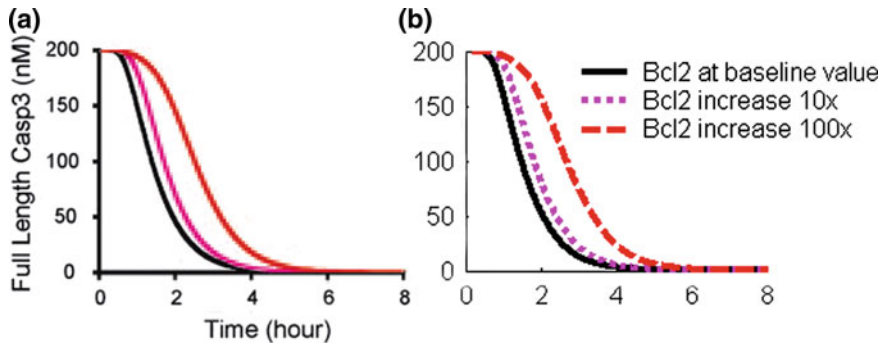
#### 6.4.4 Bcl-2's Effects on the Type II Pathway

Next, we analyzed the Caspase-3 activation kinetics by considering the different mechanisms through which it has been suggested that Bcl-2 blocks the type II pathway. In [8], [62], and [85] the authors suggested that Bcl-2 might bind with (a) Bax, (b) Bid, (c) tBid, or (d) both Bax and tBid to block the mitochondrial pathway. We implemented four different sets of rules to test each Bcl-2 binding mechanisms. We refer the interested reader to [34] for the details of the rules.

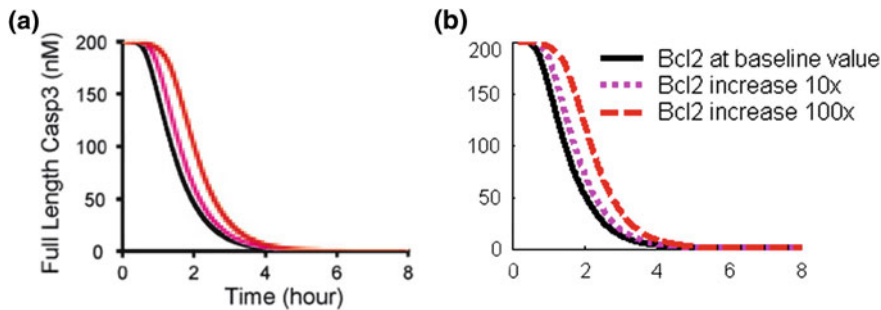
The dynamics of Caspase-3 activation were studied by increasing the baseline Bcl-2 concentration by 10-fold and 100-fold. The conclusion of [32] is that Bcl-2 binding to both Bax and tBid (d) is the most efficient mechanism for inhibiting apoptosis. Our Membrane System agrees with the observations from [32]. The results of (d) are illustrated in Fig. 6.6, and (a)–(c) can be seen in Figs. 6.8, 6.9, 6.10. A comparison of (a)–(d) at baseline Bcl-2 concentration is shown in Fig. 6.11.



**Fig. 6.8** The effects of Bcl-2 binding to Bax only. **a** The results of the solution to the system of ordinary differential equations and **b** the results of the NWT algorithm

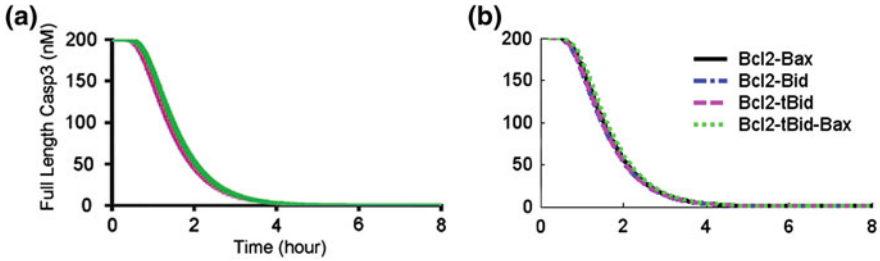


**Fig. 6.9** The effects of Bcl-2 binding to tBid only. **a** The results of the solution to the system of ordinary differential equations and **b** the results of the NWT algorithm



**Fig. 6.10** The effects of Bcl-2 binding to Bid only. **a** The results of the solution to the system of ordinary differential equations and **b** the results of the NWT algorithm





**Fig. 6.11** The results of baseline Bcl-2 concentration with each of the four mechanisms for Bcl-2 inhibition (binding with Bax only, Bid only, tBid only, or Bax and tBid). **a** The results of the solution to the system of ordinary differential equations and **b** the results of the NWT algorithm

### 6.4.5 Modeling the Behavior of the Type I Pathway

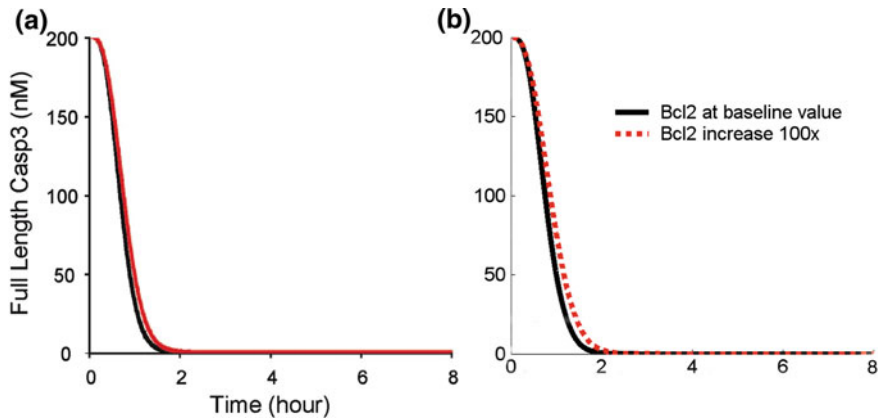
Some cells are not sensitive to Bcl-2 over expression, as described in [75]. In these cells, Caspase-3 is activated through the type I pathway, bypassing the role of the mitochondria and Bcl-2. Scaffidi et al. have suggested in [75] that the type of pathway is chosen based on the concentration of Caspase-8 generated in active form following the binding of Fas ligand to its receptor site. High concentration of active Caspase-8 allows for direct activation of Caspase-3 (type I), but if the concentration of Caspase-8 is sufficiently low, amplification of the death signal through the mitochondria is required to induce cell death (type II). We tested this hypothesis by increasing the initial concentration of Caspase-8 by 20-fold (from 33.33nMs to 666.6nMs), which was expected to lead to increased active Caspase-8\* throughout the simulation run.

We ran two different versions of the increased Caspase-8 model, using the baseline concentration of Bcl-2 and an increase of Bcl-2 by 100-fold, in order to gauge the sensitivity of the type I pathway to Bcl-2 upregulation. Fig. 6.12 shows that Caspase-3 activation was not sensitive to the increase in Bcl-2 concentration, which is the hallmark for type I pathway dominant behavior.<sup>5</sup>

Our Membrane System has yielded results comparable to the solutions to the system of ordinary differential equations. The six distinct simulations show similar apoptotic behavior to the deterministic results (Fig. 6.6 through Fig. 6.12). However, albeit the activation of Caspase-3 is similar between the two techniques, the molecular interactions throughout are different. We have compared the results of our simulator with the experimental results in [32], the deterministic results from the same paper, and the stochastic approach described in [9]. The Caspase-3 results are as expected, but the activation of Caspase-8 raises our interest. See Fig. 6.13 for a comparison between the techniques.

The experimental data and deterministic results were obtained from Fei Hua et al. We see that the decrease of full length Caspase-3 is similar in all three simulation

<sup>5</sup> For these simulations Bcl-2 was allowed to bind to both Bax and tBid, which was shown above to be the most efficient mechanism for Bcl-2 inhibition of apoptosis.



**Fig. 6.12** Investigation of the effects of Bcl-2 increase (100-fold) for the type I pathway. **a** The results of the solution to the system of ordinary differential equations and **b** the results of the NWT algorithm

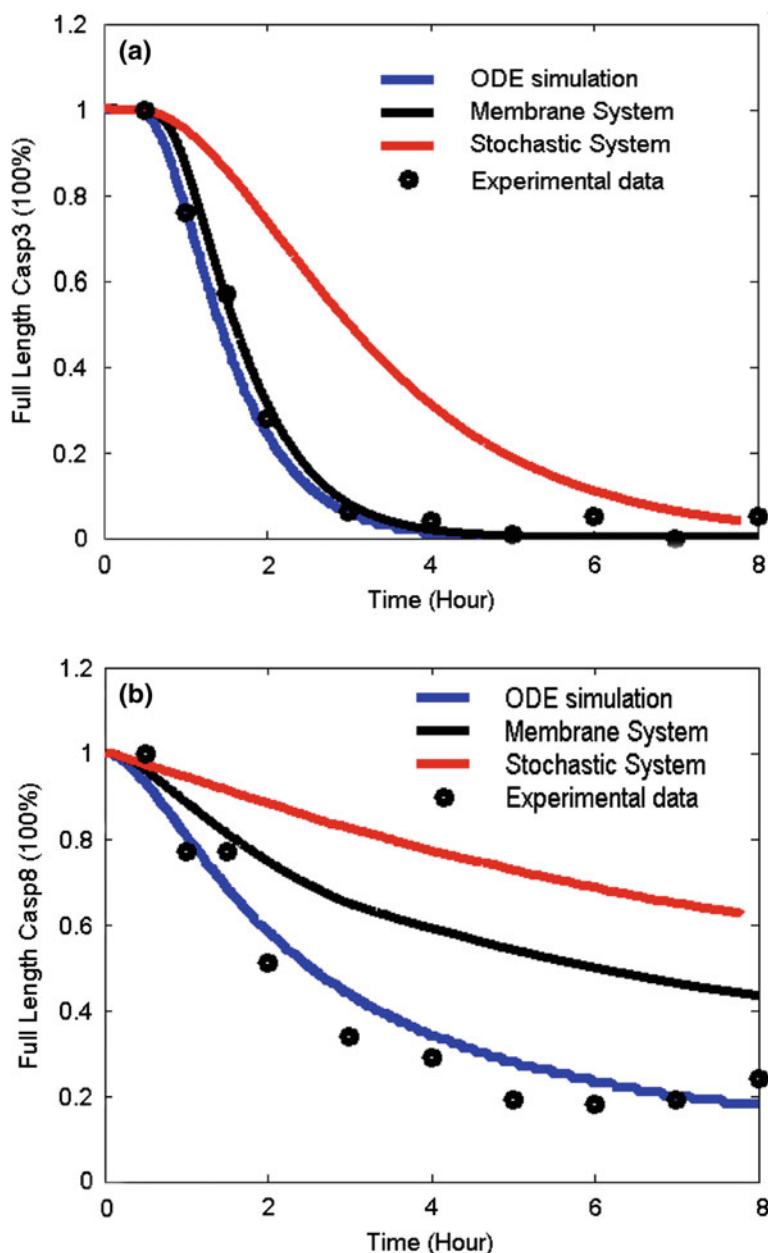
results. Interestingly, the decline of full length Caspase-8 is less prominent in the two Membrane System simulations. The contrast could be the result of the discrete nature of the Membrane Systems. As for both results being different than the experimental data, we believe that further investigation of kinetic rates of the reactions will allow for better agreement between simulation and experimentation.

#### 6.4.6 Summary for the FAS Simulation

We have chosen to simulate Fas-induced apoptosis because it has one of the most detailed descriptions/characterization in the literature (due in large part to its role in cancer and HIV research). In the interest of comparing our Membrane System with the solutions to the system of ordinary differential equations, we have implemented 101 different rules working on 53 distinct proteins and protein complexes. The pathway begins with the stimulation of FASL and ends with the activation of the effector Caspase-3. Fei Hua et al., in [32], provide the deterministic results, as well as some experimental data (from the Jurkat cell line), which they used to fit their model.

The consistency between the framework and the experimental results of [32] validates our model. Our NWT algorithm shows that Membrane Systems are an alternative to ordinary differential equations methods. We have argued that the discrete nature of our technique might be better for simulating the evolution of systems involving low numbers of molecules.

In Sect. 6.5, we will build on the rules for Fas-mediated apoptosis discuss in this section. There are a few proteins encoded in the HIV genome, which seem to have severe consequences on Fas-induced apoptosis [76]. The so-called ‘latently’ infected T cells are especially interesting in the potential strategies for the eradication of the AIDS epidemic [26].



**Fig. 6.13** Results of the three simulation techniques and experimental data provided by [32], showing decline of **a** full length Caspase-3 and **b** full length Caspase-8

## 6.5 HIV-1 Effects on the FAS Pathway

In Sect. 6.5 we will explore the qualities of the human immunodeficiency virus (HIV) which help it remain remarkably difficult to cure. Our goal is to model the effects of HIV-1 proteins on the Fas-induced apoptotic pathway. This is the first effort of its kind, and we believe it will provide further insight into HIV research and modeling.

The virus has several remarkable qualities: (1) it predominantly **infects** the cells of the **immune system**; (2) it shows a **high genetic variation** throughout the infection in a single individual due to the high error rate in the reverse transcription; (3) it **induces apoptosis** in the so-called bystander immune cells; and (4) normal immune system function can cause some HIV-infected T cells to become **latent**, entering a reversibly nonproductive state of infection. Since the latent cells are transcriptionally silent, they are virtually indistinguishable from the uninfected cells. Also, the number of latently infected cells is relatively small, which makes the experimental study of these cells difficult – current technology in biochemistry requires large numbers of the molecules/cells to be studied. It is widely believed that the latently infected CD4+ T cells represent the last barrier to an HIV cure. This Sect. 6.5 is based on our publication in *WMC09*, presenting a first modeling effort for the Fas-mediated apoptosis (or programmed cell death) of latently infected T cells [36].

We will focus on the apoptotic modeling (quality 3 for viruses as given above), since it is the avenue through which the virus destroys the effectiveness of the host's immune system. We will base our model on the work described in Sect. 6.4, using the Nondeterministic Waiting Time algorithm discussed in Sect. 6.2.1. Furthermore, in order to make the modeling effort easier and due to the high genetic variability (reason 2) of the viral genome, we will combine several similar processes together into single reactions. The kinetic constants for the new reactions, modeling the biochemical interactions involving viral proteins with the host cell, will be obtained by fitting the model to reported experiments on the infected, non-latent cells. Finally, we will simulate the reactivation of latently infect T cells by making some adjustments to the appropriate initial conditions of the system.

### 6.5.1 A Brief History of HIV

HIV, which is responsible for the onset of acquired immune deficiency syndrome (AIDS), has lead to more deaths than nearly any other virus in human history. Indeed, AIDS is called a global pandemic by the World Health Organization (WHO), and is “undoubtedly the definining public-health crisis of our time” [78]. According to statistics from the WHO, there were 33.2 million people living with HIV in 2007, 2.5 million newly infected individuals, and 2.1 million AIDS deaths [95].

When AIDS was first labeled as a diseases in 1981, there was an initial debate over what was causing the immune syndrome. It was suggested that a retrovirus could be the cause. However, retroviruses were a relatively new field of study and only a

few had yet been identified. Two labs, Gallo's in the United States and Montagnier's in France, are both credited with the discovery of HIV [2, 52]. Although political tensions surrounded the discovery, the two groups essentially agree that it was a combined effort to discover the virus behind AIDS.

There are two important qualities of HIV which makes it difficult to find a cure: (a) a high genetic variability and (b) an ability to go silent (so-called HIV latency).

One of the reasons the cure for HIV remains elusive is the high genetic variability of the virus. There are two strains of HIV: type 1 and type 2. HIV-1 is the virus whose discovery was discussed above. HIV-2 was specifically isolated [12] in West Africa in 1986. Unlike HIV-1, the type 2 strain remains confined to West Africa [74]. Since HIV-1 is more virulent and transmissible [74], the discussion and modeling efforts of this section will be concerned with the type 1 variant.

HIV-1 can be broken down into multiple subtypes. Infected individuals are susceptible to co-infections and superinfections [78]; this can lead to new recombinant forms of the virus. For instance, in Southeast Asia an estimated twenty percent of infections come from recombinant forms [78]. Lack of immunization and continuous evolution of the viral genome makes vaccine development a considerable challenge. The modeling community must remain aware of the different subtypes, to avoid ill-fit models based on these genetically distinct subtypes, resulting in poor approximation of reality.

### 6.5.2 *AIDS Pathogenesis*

The pathogenesis of AIDS is attributed to the depletion of the host's CD4+ T cells, the loss of which results in a dysfunctional immune system. Finkel and colleagues in [15] concluded that HIV-1 infection causes death predominantly in the so-called bystander T cells. These healthy, uninfected cells are marked for destruction by the neighboring HIV-1-infected cells. The mechanism of the bystander cell death was shown to be apoptosis. Proteins encoded by the HIV-1 genome exhibit anti- and pro-apoptotic behavior on infected and bystander cells, enhancing or inhibiting a cell's ability to undergo apoptosis.

There are numerous drugs available for limiting the impact of HIV-1 on the immune system; the most successful approach, highly active anti-retroviral therapy (HAART), is a combination of several types of drugs, targeting different mechanisms of HIV-1 infection and proliferation. Although HAART has proven to be effective in the reduction or elimination of viremia [66], it is ineffective in the complete eradication of the viral infection. The HIV-1 infection is able to persist in a dormant state throughout the entire time a patient is on HAART. The way this is accomplished is one of the most remarkable qualities of HIV—i.e., latency.

Latent reservoirs of HIV-1 have been detected in HIV-1-infected patients [10, 11]. Latently infected cells are relatively rare—about 1 in  $10^6$  resting T cells [11]. However, they are considered to be the largest obstacle in combating HIV-1 infection

[16, 77, 80]. Understanding the mechanisms behind HIV-1 latency is a focal point for current AIDS-related research (for a recent review on latency see [26]).

There are two types of latency described in the literature. The first, preintegration latency, refers to resting T cells containing unintegrated HIV-1 DNA. If a T cell is in a resting state, the HIV-1 DNA is not able to quickly integrate into the host's viral genome. Since the unintegrated HIV-1 DNA is labile and reverse transcription of HIV-1 RNA is slow (on the order of days) [67, 91, 92, 94], it is believed that patients with reduced viremia after several months of HAART therapy do not have resting T cells with unintegrated HIV-1 DNA [5]. Hence, we will not concern ourselves with modeling preintegration latency. We will instead discuss the second form of latency – postintegration latency. It is this alternative form of latency which will be the focus of the rest of the section.

Postintegration latency refers to resting T cells with stably integrated HIV-1 DNA. These cells can provide a reservoir for viral reproduction for years [16]. The cells exist as a natural consequence to normal healthy immune function. When the body is invaded by an organism, T cells are activated to destroy the invading pathogen. Once the pathogen is destroyed, many of the T cells commit apoptosis or else they would persist in killing other cells at the inevitable detriment of the host organism. However, a few of these active T cells return to a quiescent state. This return to a nonactive status is the basis for so-called memory T cells.

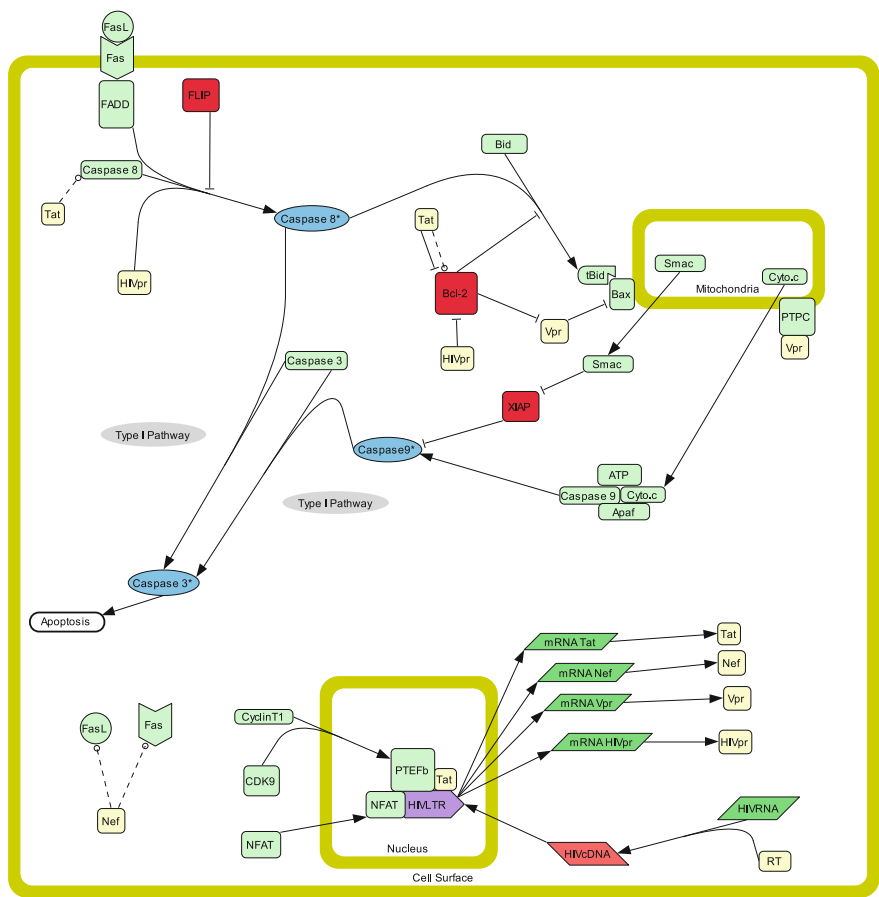
When an activated HIV-1-infected T cell turns into a memory T cell, this is very troublesome for the infected individual. The individual now has a T cell which is virtually indistinguishable from all of the other resting T cells, and yet it is infected with the HIV-1 genome. The cell can persist almost indefinitely in this state. Hence, when the individual goes off HAART and the resting HIV-1-infected T cell is reactivated, viraemia is quickly restored and the individual will succumb to AIDS.

It is because of their long lifespan and ability to restore viraemia that we have chosen to model the reactivation of a postintegration latently infected CD4+ T cell. We chose to model the Fas-induced apoptosis of these cells because the effects are well characterized in the literature and no one has made an attempt to do so before.

As far as we know, this paper reports the first attempt at modeling the Fas-mediated apoptotic signaling pathway in reactivated latently infected CD4+ T cells. We will draw upon the system we laid out in Sect. 6.4, which is based on information for the Jurkat T cell line from [32] (and references therein). We have extended the model from Sect. 6.4 in order to better understand the reactivation of latently infected T cells.

### 6.5.3 HIV-1 Infection

There is still some debate about the effects of HIV-1 proteins on cellular signaling networks; however, we have pooled the collective knowledge of the biological community in order to categorize and model the described functions of various HIV proteins. For an illustration of the Fas pathway and the involvement of the HIV proteins we refer the reader to Fig. 6.14. We will explain the inspiration for the model below.



**Fig. 6.14** A picture of the model for HIV-1 protein effects on Fas signaling. The activation of Caspase 3 is the end of the signaling cascade—irrevocably leads to cell death. The type I pathway involves direct activation of Caspase 3 by Caspase 8\*. The type II pathway requires signal amplification by way of the mitochondria, resulting in the activation of Caspase 3 by Caspase 9\*. The HIV-1 Tat protein upregulates inactive Caspase 8 and Bcl-2, but it can also downregulate Bcl-2. Vpr upregulates Bcl-2 and downregulates Bax. HIV Protease can cleave Bcl-2 into an inactive form and it can also cleave Caspase 8 into active Caspase 8. Finally, Nef protein upregulates Fas ligand and Fas receptor

The mechanisms behind HIV-1 infection of CD4+ T cells are well understood. A spike on the virus, the gp120 envelope glycoprotein, binds to the CD4 receptor of the target cell and, in conjunction with subsequent binding to a coreceptor (CCR5 or CXCR4), a path is opened for the virus to inject its contents into the cell [7, 88]. Reverse transcriptase creates cDNA from the HIV-1 RNA and the genome of the virus is implanted into the cell's own DNA for future production. During this time, the immune system fails to detect and destroy the infected cell.

Upon infection, the contents of the virion (e.g., Vpr, HIV protease ( $HIV_{pr}$ ), reverse transcriptase (RT), and HIV RNA ( $HIV_{RNA}$ )) are released into the cytoplasm [6]. In the newly infected and active CD4+ T cells, the  $HIV_{RNA}$  is converted to cDNA ( $HIV_{cDNA}$ ) by the reverse transcriptase about five hours post-infection [42]. The  $HIV_{cDNA}$  is then integrated into the host's genome with the help of the viral integrase approximately one hour later [17]. We will formalize these rules as we discuss the inspiration from the literature behind their creation. For our convenience, we have labeled the integrated HIV genome as  $HIV_{LTR}$  in our rules.  $HIV_{LTR}$  is the basis for interactions involving the HIV long terminal repeat; in our model, it is a necessary component for all reactions pertaining to HIV-1 protein production.

After integration of the viral DNA, gene expression of HIV proteins becomes possible. The nuclear factor of activated T cells (NFAT) and NF- $\kappa$ B have been shown to play important roles in HIV gene expression [43, 54]. In a resting CD4+ T cell, NF- $\kappa$ B is sequestered in the cytoplasm by its inhibitor, I $\kappa$ B. Following cellular activation, NF- $\kappa$ B is released by its inhibitor, which allows it to relocate to the nucleus where it can bind to the  $HIV_{LTR}$ . Also following T cell activation, NFAT, located in the cytoplasm of resting CD4+ T cells, undergoes dephosphorylation and translocation to the nucleus where it can bind to the  $HIV_{LTR}$  [43]. Once NF- $\kappa$ B and NFAT are translocated to the nucleus, they can bind to the  $HIV_{LTR}$ , combining their efforts to synergistically enhance the promoter activity. Moreover, [43] shows that the combined effects of Tat, NF- $\kappa$ B and NFAT is much stronger than the pairings of Tat and NF- $\kappa$ B or Tat and NFAT. In our model, we have combined the roles of NF- $\kappa$ B and NFAT. The translocation and binding rules for NFAT (and NF- $\kappa$ B) are shown in Table 6.5.

Multiply spliced (MS) HIV-1 mRNAs—responsible for Tat/Rev protein creation—are detectable in resting CD4+ T cells [47]. However, due to the inefficient export of the mRNA transcripts to the cytosol, Tat and Rev proteins are undetectable in the latent cells. Activation of these latent cells leads to production of Tat and Rev, and subsequent upregulation of all HIV-1 proteins. In order for the infected cells to create HIV proteins other than Tat and Rev, the transcriptional elongation induced by Tat and the efficient nuclear export of MS HIV-1 RNAs by Rev are required. Our latent cell model, beginning with cellular activation, initially allows for inefficient creation of Tat proteins. We chose not to model Rev, since it has no known Fas apoptotic function; its exporting functions are incorporated into the kinetic constants governing mRNA translocation. Once Tat is located in the nucleus, it requires the help of two other proteins provided by the host cell: CyclinT1 and CDK9.

In an inactivated cell, CyclinT1 and CDK9 are sequestered in the cytoplasm [55]. Upon T cell activation, they are relocated to the nucleus. CyclinT1 and CDK9 combine to make up the positive-acting transcription elongation factor (P-TEFb) complex. The binding of P-TEFb and Tat at the  $HIV_{LTR}$  allows the hyperphosphorylation of RNA polymerase II (RNAPII), resulting in increased transcriptional elongation. The translocation and binding rules for CyclinT1, CDK9 and Tat are formalized in Table 6.5. The transcription, translocation, and translation rules involving HIV-1 mRNA molecules are also summarized in Table 6.6.



### 6.5.4 HIV-1-Related Effects on the Fas Pathway

Aside from its role in transcriptional elongation, the Tat protein is responsible for both pro- and anti-apoptotic behavior. In [4], the authors demonstrated that increased Tat expression causes upregulation of inactive Caspase-8, which is a pro-apoptotic molecule. Also, Tat has been associated with the downregulation of Bcl-2 [76], which is an anti-apoptotic molecule. Given the pro- and anti-apoptotic duties of Caspase-8 and Bcl-2, respectively, it would appear that a cell with high levels of Tat has increased susceptibility to apoptosis. Conversely, [15] claims that Tat upregulates Bcl-2, resulting in decreased apoptotic rates of cells. Tat has also been implicated in the upregulation of Fas ligand on the cell surface [4, 89], which may effect the cell through autocrine signaling. The anti- and pro-apoptotic rules for Tat are found in Table 6.7.

The HIV-1 Vpr has been shown to both enhance and inhibit the Fas signaling cascade. Upon infection, the  $\sim 700$  molecules of Vpr in the virion are injected into the cytoplasm of the cell [6]. At low levels, Vpr has been shown to prohibit apoptosis by upregulating Bcl-2 and downregulating Bax [13]. However, higher concentrations of Vpr affect the mitochondrial membrane permeability via interactions with the permeability transition pore complex (PTPC), resulting in the release of Cytochrome c into the cytoplasm [38]. In the same paper, the authors also demonstrated that Bcl-2 can inhibit the effects of Vpr on the PTPC. The various apoptotic roles of Vpr we define in Table 6.8.

Another protein packaged in HIV-1 virions, HIV<sub>pr</sub>, plays an important role in the Fas pathway. The HIV<sub>pr</sub> has been shown to cleave Bcl-2 into a deactivated state [81], while it also cleaves Caspase-8 [57] into active form. Both rules are pro-apoptotic and are in Table 6.9.

Finally, we define two pro-apoptotic rules for the Nef protein. Zauli et al. discovered in [93] that Nef can play a role in cell death by upregulating Fas receptor and Fas ligand on the cell surface. Upregulating the receptor sites of Fas on the cell surface prepares the cell for ligand binding, and can initiate the Fas-induced apoptotic signaling cascade. The upregulation of Fas ligand may protect the infected cell from cytotoxic T cells, or it could be part of autocrine signaling. The three rules for upregulation and translocation of Fas and Fas ligand are in Table 6.10.

### 6.5.5 Modeling Results

We added all of the rules from Tables 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, 6.10 to the Fas model described in Sect. 6.4—interested reader should also consult [34] for the complete list. From this, we are able to simulate two types of cells: *nonlatent* and *latent*. The differences between the two models are the initial protein multiplicities. The *nonlatent* cell is an activated T cell which has just been infected with the contents of the HIV-1 virion. The HIV-1 RNA and other viral proteins are in the cytoplasm.

**Table 6.4** Reactions involving translocation of the HIV genome and integration into the host's genome

Reaction	Reaction rate
1: $\text{HIV}_{RNA} + \text{RT} \rightarrow \text{HIV}_{cDNA} + \text{RT}$	$k_{21}$
2: $\text{HIV}_{cDNA} \rightarrow \text{HIV}_{cDNA}$ (nuclear import)	$k_{22}$
3: $\text{HIV}_{cDNA} \rightarrow \text{HIV}_{LTR}$	$k_{22}$

**Table 6.5** Reactions involving HIV long terminal repeat (LTR) and HIV mRNA production

Reaction	Reaction Rate
4: $\text{NFAT} \rightarrow \text{NFAT}$ (nuclear import)	$k_{23}$
5: $\text{CDK9} \rightarrow \text{CDK9}$ (nuclear import)	$k_{24}$
6: $\text{CyclinT1} + \text{CDK9} \rightarrow \text{PTEFb}$	$k_{25}$
7: $\text{NFAT} + \text{HIV}_{LTR} \rightarrow \text{HIV}_{LTR}:\text{NFAT}$	$k_{26}$
8: $\text{HIV}_{LTR}:\text{NFAT} + \text{Tat} \rightarrow \text{HIV}_{LTR}:\text{NFAT}:\text{Tat}$	$k_{27}$
9: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat} + \text{PTEFb} \rightarrow \text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb}$	$k_{28}$
10: $\text{HIV}_{LTR} \rightarrow \text{HIV}_{LTR} + \text{mRNA}_{Tat}$	$k_{29}$
11: $\text{HIV}_{LTR} \rightarrow \text{HIV}_{LTR} + \text{mRNA}_{Vpr}$	$k_{29}$
12: $\text{HIV}_{LTR} \rightarrow \text{HIV}_{LTR} + \text{mRNA}_{HIVpr}$	$k_{29}$
13: $\text{HIV}_{LTR} \rightarrow \text{HIV}_{LTR} + \text{mRNA}_{Nef}$	$k_{29}$
14: $\text{HIV}_{LTR}:\text{NFAT} \rightarrow \text{HIV}_{LTR}:\text{NFAT} + \text{mRNA}_{Tat}$	$k_{30}$
15: $\text{HIV}_{LTR}:\text{NFAT} \rightarrow \text{HIV}_{LTR}:\text{NFAT} + \text{mRNA}_{Vpr}$	$k_{30}$
16: $\text{HIV}_{LTR}:\text{NFAT} \rightarrow \text{HIV}_{LTR}:\text{NFAT} + \text{mRNA}_{HIVpr}$	$k_{30}$
17: $\text{HIV}_{LTR}:\text{NFAT} \rightarrow \text{HIV}_{LTR}:\text{NFAT} + \text{mRNA}_{Nef}$	$k_{30}$
18: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat} \rightarrow \text{HIV}_{LTR}:\text{NFAT}:\text{Tat} + \text{mRNA}_{Tat}$	$k_{31}$
19: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat} \rightarrow \text{HIV}_{LTR}:\text{NFAT}:\text{Tat} + \text{mRNA}_{Vpr}$	$k_{31}$
20: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat} \rightarrow \text{HIV}_{LTR}:\text{NFAT}:\text{Tat} + \text{mRNA}_{HIVpr}$	$k_{31}$
21: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat} \rightarrow \text{HIV}_{LTR}:\text{NFAT}:\text{Tat} + \text{mRNA}_{Nef}$	$k_{31}$
22: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} \rightarrow$ $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} + \text{mRNA}_{Tat}$	$k_{32}$
23: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} \rightarrow$ $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} + \text{mRNA}_{Vpr}$	$k_{32}$
24: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} \rightarrow$ $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} + \text{mRNA}_{HIVpr}$	$k_{32}$
25: $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} \rightarrow$ $\text{HIV}_{LTR}:\text{NFAT}:\text{Tat}:\text{PTEFb} + \text{mRNA}_{Nef}$	$k_{32}$

The HIV-1 RNA must be incorporated into the host's genome before the viral protein production can begin. The *latent* model is a newly activated T cell with no HIV-1 proteins present. However, the HIV-1 genome is already integrated into the host's DNA.

As we have discussed earlier, the *nonlatent* cell is used for the model fitting, since the majority of information about HIV-1 proteins pertains to these types of cells. The reason for this lies in the fact that latent cells are transcriptionally silent (virtually undetectable) and relatively rare. For instance, in Fig. 6.15a, the results from the *nonlatent* simulation show the activity of Tat in that full length (inactive) Caspase-8 increases by a factor of three. Our simulation agrees with the observations

**Table 6.6** Translation and degradation rules for HIV mRNA

Reaction	Reaction Rate
26: $\text{mRNA}_{Tat} \rightarrow \text{mRNA}_{Tat}$ (nuclear export)	$k_{33}$
27: $\text{mRNA}_{Nef} \rightarrow \text{mRNA}_{Nef}$ (nuclear export)	$k_{33}$
28: $\text{mRNA}_{Vpr} \rightarrow \text{mRNA}_{Vpr}$ (nuclear export)	$k_{33}$
29: $\text{mRNA}_{HIVpr} \rightarrow \text{mRNA}_{HIVpr}$ (nuclear export)	$k_{33}$
30: $\text{mRNA}_{Tat} \rightarrow \text{mRNA}_{Tat} + \text{Tat}$	$k_{34}$
31: $\text{mRNA}_{Nef} \rightarrow \text{mRNA}_{Nef} + \text{Nef}$	$k_{34}$
32: $\text{mRNA}_{Vpr} \rightarrow \text{mRNA}_{Vpr} + \text{Vpr}$	$k_{34}$
33: $\text{mRNA}_{HIVpr} \rightarrow \text{mRNA}_{HIVpr} + \text{HIV}_{pr}$	$k_{34}$
34: $\text{mRNA}_{Tat} \rightarrow \text{degraded}$	$k_{35}$
35: $\text{mRNA}_{Nef} \rightarrow \text{degraded}$	$k_{35}$
36: $\text{mRNA}_{Vpr} \rightarrow \text{degraded}$	$k_{35}$
37: $\text{mRNA}_{HIVpr} \rightarrow \text{degraded}$	$k_{35}$

**Table 6.7** Reactions involving Tat protein

Reaction	Reaction Rate
38: $\text{Tat} \rightleftharpoons \text{Tat}$ (nuclear import/export)	$k_{36f}, k_{35r}$
39: $\text{Tat} \rightarrow \text{Tat} + \text{Casp8}$	$k_{37}$
40: $\text{Tat} \rightarrow \text{Tat} + \text{Bcl2}$	$k_{38}$
41: $\text{Tat} \rightarrow \text{FasL} + \text{Tat}$	$k_{39}$
42: $\text{Tat} + \text{Bcl2} \rightarrow \text{Tat}$	$k_{40}$

**Table 6.8** Reactions involving Vpr protein

Reaction	Reaction Rate
43: $\text{Vpr} + \text{Bax} \rightarrow \text{Vpr}$	$k_{42}$
44: $\text{Vpr} + \text{Bcl2} \rightarrow \text{Vpr:Bcl2}$	$k_{43}$
45: $\text{Vpr:Bcl2} \rightarrow \text{Vpr} + \text{Bcl2}$	$k_{44}$
46: $\text{Vpr} + \text{PTPC} \rightarrow \text{Vpr:PTPC}$	$k_{45}$
47: $\text{Vpr:PTPC} + \text{Cyto.c} \rightarrow \text{Cyto.c}^* + \text{Vpr:PTPC}$	$k_{46}$

**Table 6.9** Reactions involving HIV protein

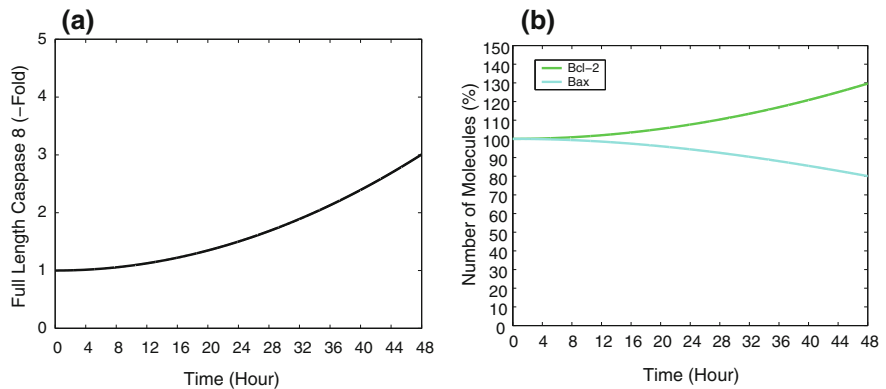
Reaction	Reaction Rate
48: $\text{HIV}_{pr} + \text{Casp8} \rightarrow \text{HIV}_{pr} + \text{Casp8}^*$	$k_{47}$
49: $\text{HIV}_{pr} + \text{Bcl2} \rightarrow \text{HIV}_{pr}$	$k_{48}$

of [4]. Also, in Fig. 6.15b, our model shows Vpr-induced upregulation of Bcl-2 and downregulation of Bax by 30 and 20 %, resp. Our results agree with the experimental results on Vpr described in [13].

We will next consider the activation of Caspase-3. In Fig. 6.16, both the *nonlatent* and *latent* models are shown to exhibit the onset of apoptosis—total activation of Caspase-3—after approximately two days. Our results indicate that reactivated latently infected CD4+ T cells activate all of the Caspase-3 molecules earlier than

**Table 6.10** Reactions involving Nef protein

Reaction	Reaction Rate
50: Nef → Nef + Fas	$k_{49}$
51: Nef → Nef + FasL	$k_{50}$
52: FasL → FasL (to cell surface)	$k_{51}$



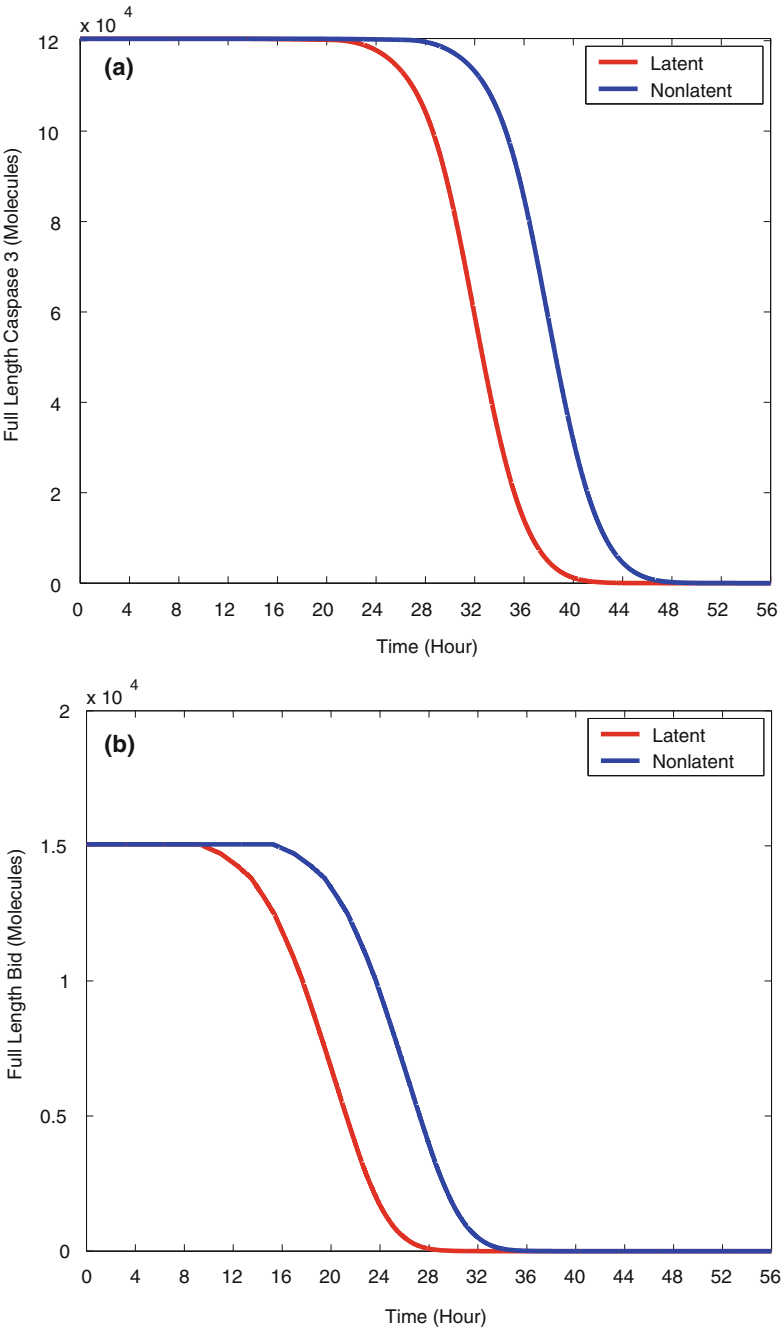
**Fig. 6.15** **a** Tat Protein Upregulates Caspase-8 levels by three-fold. **b** Vpr upregulates Bcl-2 and downregulates Bax by 30 and 20%, resp

the *nonlatent* model. Also, in Fig. 6.16, we show the truncation of Bid, which is a necessary step in the induction of the type II pathway. Active Caspase-8 is responsible for the truncation of Bid, so we are seeing the downstream effects of Caspase-8 activation.

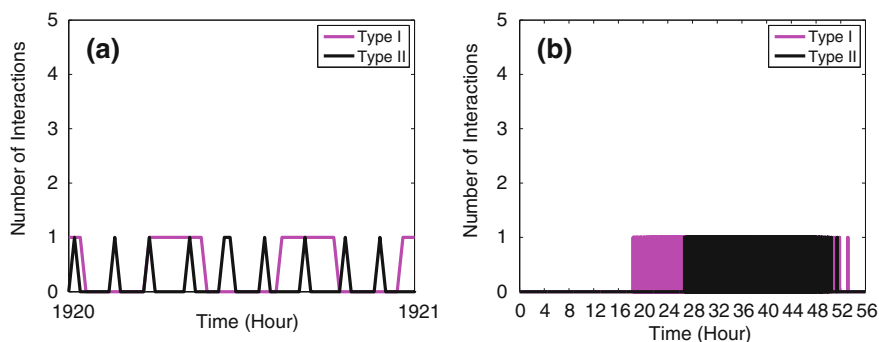
Next, let us consider the mechanisms behind Caspase-3 activation in the *latent* and *nonlatent* models. According to the rules in Appendix A, an interaction between full length Caspase-3 and active Caspase-8 or Caspase-9 can have two outcomes: the activation of Caspase-3 or not. Both of our models show cooperation between the two pathways, which is not explicitly stated in the literature. The *nonlatent* results (Fig. 6.17) show the first interactions between Caspase-3 and Caspase-8\* molecules occur just after 18h into the run. It isn't until ~10h later (26h into the run) that we begin to see Caspase-3 interactions with Caspase-9\*, after signal amplification through the mitochondria. As discussed in [32, 35], given a sufficiently high initial concentration of Caspase-8 in the cell, signal amplification is not necessary to induce apoptosis. For this model, we set the initial level of Caspase-8 to be insufficient for apoptosis by the type I pathway.

The results of the *latent* simulation are similar to the *nonlatent*, where both pathways appear to govern Caspase-3 activation. In the *latent* run (Fig. 6.18), we see type I interactions first occur about 12h into the simulation, while type II molecular binding occurs after 21 h.

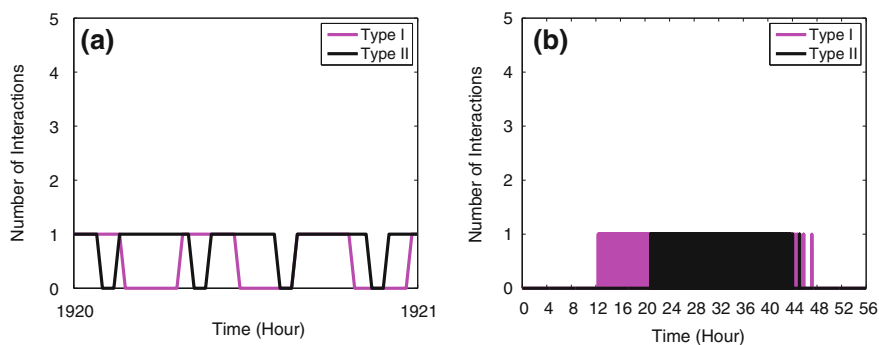
Although Figs. 6.17b and 6.18b imply type I interactions occur more frequently than type II, it must be noted that, due to the kinetics governing these binding



**Fig. 6.16** **a** Total reduction of full length Caspase-3 is seen after ~40 h in the *latent* model, whereas the *nonlatent* model takes ~47 h. **b** The decline of Bid through interactions with Caspase-8, leading to a rise in tBid



**Fig. 6.17** Interactions involved in the type I and type II pathways for the *latent* simulation. **a** The results for the three days of simulation and **b** An excerpt of one minute from the three day simulation (from 32 to 32h and 1 min)



**Fig. 6.18** Interactions involved in the type I and type II pathways for the *nonlatent* simulation. **a** The results for the three days of simulation and **b** An excerpt of one minute from the three day simulation (from 32 to 32h and 1 min)

rules, Caspase-8\* can remain bound to Caspase-3 for a longer period of time than Caspase-9\*. Therefore, although it seems that Caspase-8\* binds to Caspase-3 more frequently, the reactions are merely slower. In fact, both models exhibit more interactions between Caspase-9\* and Caspase-3.

### 6.5.6 Summary for Simulating HIV Latency

Based on the biological evidence in the literature, we constructed a model for the effects of HIV-1 proteins on the Fas-mediated apoptosis pathway. This work is the first of its kind, simulating Fas-induced apoptosis in reactivated latently infected CD4+ T cells. We have provided some preliminary results in an effort to understand CD4+ T cell latency. Interestingly, our results show a cooperation between the type

I and type II pathways. We have not been able to verify an explanation for this in the available literature.

Finally, we would like to note that the experimental information on the latent HIV-1-infected CD4+ T cells is scarce, due to the fact that these cells are found in such small numbers *in vivo*. Therefore, our model relies heavily on applying the knowledge of activated HIV-1-infected CD4+ T cells. We look forward to new experimental results about these enigmatic cells, which we will use to refine the model.

## 6.6 Conclusions and Final Remarks

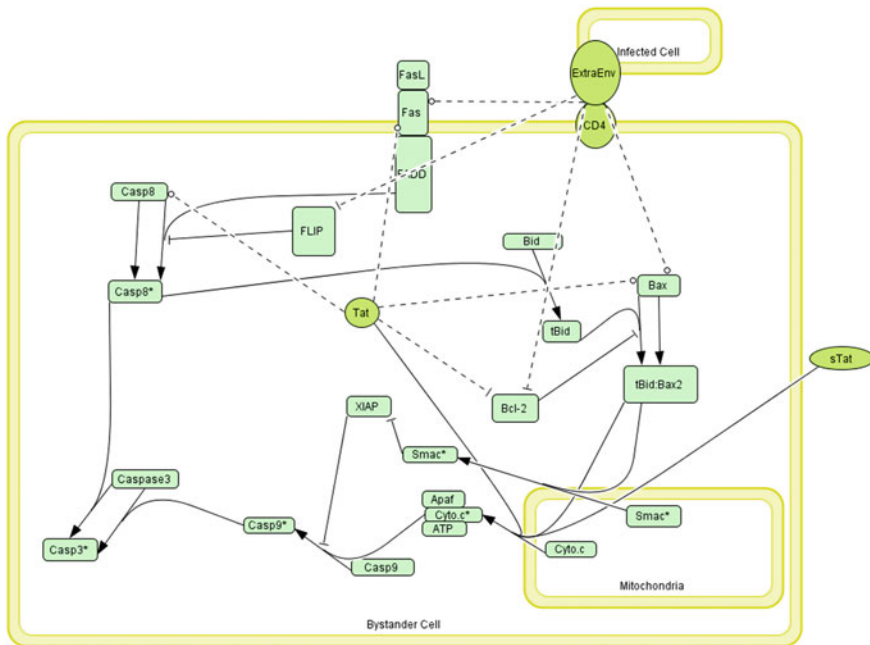
Using our NWT algorithm, we have shown results for modeling some popular networks—e.g., the Lotka-Volterra predator-prey model and a circadian rhythm model. With these two models, we have shown our algorithm can exhibit Gillespie-like results with the latter, while NWT simulations of the former model show similarities to the results of the solution to the system of ordinary differential equations. However, the computational load of the nondeterministic decisions in our NWT algorithm is far less than the Gillespie Algorithm, as is the case in the noise-induced oscillations for the circadian rhythm model.

We have used our NWT algorithm to explore Fas-induced apoptosis. We started by simulating a model developed by the Lauffenberger lab at M.I.T. [32]. Our simulator is capable of showing comparable results to the solution of the system of ordinary differential equations from that group. However, when compared with the experimental results, we did notice some differences between our discrete, nondeterministic technique, the ordinary differential equations, and experimental results provided by the M.I.T. lab. Activation of Caspase 3 was very close, but Caspase 8 was a bit different. In other words, the end of the signaling cascade occurred at the same time, but a critical earlier component showed different activity. We concluded that this was due to the discrete nature of our simulation.

After an extensive literature review, we were able to extend the Fas model of [32] to incorporate HIV-1 activity. This was the first attempt someone has made in modeling Fas-induced apoptosis in HIV-1-infected cells. We made a special effort to model the so-called latently infected T cells, which are considered the last barrier in the eradication of HIV-1-infection. There are some interesting directions to go with this research.

### 6.6.1 Extensions on the HIV Model

There are several avenues to explore in extending the HIV model from Sect. 6.5. For instance, we would like to model the effects of HIV-1 proteins on bystander cell apoptosis. As mentioned in the introduction, HIV-1 appears to primarily kill uninfected bystander T cells [15]. Various mechanisms have been reported for the



**Fig. 6.19** The effects of an HIV-1-infected T cell on its neighboring healthy, bystander cells

destruction of the bystander cells. Along with Fas-induced apoptosis, other possible mechanisms for bystander cell death are reviewed in [76]. Upon being exocytosed by an infected cell, several of the proteins encoded in HIV-1 can exhibit destructive qualities when interacting with neighboring bystander cells—either on the surface or through endocytosis. In Fig. 6.19, we see a proposed model for bystander cell apoptosis.

There are a few HIV-1 proteins we have ignored in this model, because they affect T cells in ways not within the scope of our current efforts. For example, soluble and membrane-bound Env can bind to the CD4 receptor of bystander cells. It was shown that ligation of the CD4 receptor by Env, is sufficient to increase apoptosis in bystander cells. The reasons for the increased apoptotic rates following Env-CD4 binding can be attributed to Bcl-2 down-regulation [30], increased Caspase 8 activation [1], and upregulation of Fas [64], FasL and Bax [76].

Modeling a cluster of cells would be a possible extension of this model. Using MPI, we can have each node of a cluster model a distinct cell. One (or more) of the nodes can be an HIV-1-infected cell, while many of the nodes can represent bystander cells. We can simulate the effects of the HIV-infected cell on the healthy, bystander cells. Besides HIV-1-related effects on the Fas-mediated apoptotic pathway, there are other directions to go with Fas modeling.



### 6.6.2 *Calcium's Role in Apoptosis*

In recent years, calcium's role in apoptosis has received increased attention. For a recent review, we refer the reader to [70]. It seems that calcium is capable of exhibiting both pro- and anti-apoptotic characteristics. While a large portion of the literature illustrates calcium's role in the intrinsic (sometimes referred to as the mitochondrial) pathway, there is also evidence showing its role in the extrinsic pathway.

We propose an exploration into the role of calcium as an apoptotic signaling molecule. The effort will combine experimental and computational techniques to derive new kinetic rates and extrapolate apoptotic signaling behavior for a variety of cell types—besides Jurkat T cells, the lab of Dr. DeCoster is also investigating neurons and astrocytes.

Calcium release from the Endoplasmic Reticulum (ER) can have profound impact on the mitochondria and, thus, the intrinsic apoptotic pathway. The “cross-talk” between the ER and the mitochondria is responsible for mediating signaling, ATP production and apoptosis [84]. The functional significance of the physical and physiological link between the ER and mitochondria is profound [24, 73]. For instance, in many apoptotic models the release of  $\text{Ca}^{2+}$  from the ER is directly responsible for mitochondrial calcium overload [25, 69]. Although  $\text{Ca}^{2+}$  has a low affinity for the mitochondrial  $\text{Ca}^{2+}$  transporters, it is the close proximity between the mitochondria and the ER which allows for the rapid accumulation of calcium in the mitochondrial matrix [68]. The interactions between  $\text{Ca}^{2+}$  and the mitochondria, can lead to a variety of mitochondrial activity. For instance,  $\text{Ca}^{2+}$  overload can result in a loss of mitochondrial membrane potential, which can lead to increased release of Cytochrome c into the cytosol.

Finally, as a tertiary research direction, we have the option of investigating calcium oscillations in relation to HIV latency. There is recent evidence supporting the idea that calpains may play a critical role in apoptosis. These cysteine proteases are activated in a  $\text{Ca}^{2+}$ -dependent manner, and they may be effective in inhibiting the activation of latent HIV-infected cells [82]. Hence, as we establish our models of  $\text{Ca}^{2+}$ -mediated apoptotic signaling, we can also work the new dynamics from the lab of Dr. DeCoster into the HIV apoptotic model we proposed in Sect. 6.5.

We believe that this research area is extremely relevant for the future when the new generation sequencing machines as well as the new methods of quantifying the protein-protein interactions (such as FRET analysis) will provide better data and bigger systems for the cellular pathways. This explosion of information will lead to the need of faster simulators based on discrete mathematics.

## References

1. A. Algeciras-Schimnich, S.R. Vlahakis, A. Villasis-Keever, T. Gomez, C.J. Heppelmann, G. Bou, C.V. Paya, CCR5 Mediates Fas- and caspase-8 dependent apoptosis of both uninfected and HIV infected primary human CD4 T cells. *AIDS* **16**(11), 1467–1478 (2002)

2. F. Barre-Sinoussi et al., Isolation of T-lymphocyte retrovirus from a patient at risk for acquired immune deficiency syndrom (AIDS). *Science* **220**(4599), 868–871 (1983)
3. R.L. Bar-Or, R. Maya, L.A. Segel, U. Alon, A.J. Levine, M. Oren, Generation of oscillations by the p53-Mdm2 feedback loop: a theoretical and experimental study, *Proc. Natl. Acad. Sci.* **97**(21), 11250–11255 (2000)
4. S.R. Bartz, M. Emerman, Human immunodeficiency virus type 1 tat induces apoptosis and increases sensitivity to apoptotic signals by up-regulating FLICE/Caspase-8. *J. Virol.* **73**(3), 1956–1963 (1999)
5. J.N. Blankson et al. Biphasic decay of latently infected CD4+ T cells in acute human immunodeficiency virus type 1 infection. *J. Infect. Dis.* **182**(6), 1636–1642 (2000)
6. J.A. Briggs, M.N. Simon, I. Gross, H.G. Kräusslich, S.D. Fuller, V.M. Vogt, M.C. Johnson, The stoichiometry of gag protein in HIV-1. *Nat. Struct. Mol. Biol.* **11**(7), 672–675 (2004)
7. D.C. Chan, P.S. Kim, HIV entry and its inhibition. *Cell* **93**(5), 681–684 (1998)
8. E.H. Cheng, M.C. Wei, S. Weiler, R.A. Flavell, T.W. Mak, T. Lindsten, S.J. Korsmeyer, BCL-2, BCL-XL sequester BH3 domain-only molecules preventing BAX- and BAK-mediated mitochondrial apoptosis. *Mol. Cell* **8**(3), 705–711 (2001)
9. S. Cheruku, A. Păun, F.J. Romero-Campero, M.J. Pérez-Jiménez, O.H. Ibarra, Simulating FAS-induced apoptosis by using P Systems. *Prog. Nat. Sci.* **17**(4), 424–431 (2007)
10. T.W. Chun, D. Finzi, J. Margolick, K. Chadwick, D. Schwartz, R.F. Siliciano, In vivo fate of HIV-1-infected T cells: Quantitative analysis of the transition to stable latency. *Nat. Med.* **1**(12), 1284–1290 (1995)
11. T.W. Chun et al., Quantification of latent tissue reservoirs and total body viral load in HIV-1 infection. *Nature* **387**(6629), 183–188 (1997)
12. F. Clavel et al., Isolation of a new human retrovirus from West African patients with AIDS. *Science* **233**(4761), 343–346 (1986)
13. L. Conti et al., The HIV-1 vpr protein acts as a negative regulator of apoptosis in a human lymphoblastoid T cell line: possible implications for the pathogenesis of AIDS. *J. Exp. Med.* **187**(3), 403–413 (1998)
14. J. Debnath, E.H. Baehrecke, G. Kroemer, Does autophagy contribute to cell death? *Autophagy* **1**(2), 66–74 (2005)
15. T.H. Finkel et al., Apoptosis occurs predominately in bystander cells and not in productively infected cells of HIV- and SIV-infected lymph nodes. *Nat. Med.* **1**(2), 129–134 (1995)
16. D. Finzi et al., Latent infection of CD4+ T cells provides a mechanism for lifelong persistence of HIV-1, even in patients on effective combination therapy. *Nat. Med.* **5**(5), 512–517 (1999)
17. C.M. Farnet, W.A. Haseltine, Determination of viral proteins present in the human immunodeficiency virus type 1 preintegration complex. *J. Virol.* **65**(4), 1910–1915 (1991)
18. L. Formigli et al., Aponecrosis: morphological and biochemical expoloration of a syncletic process of cell death sharing apoptosis and necrosis. *J. Cell. Physiol.* **182**(1), 41–49 (2000)
19. A. Funahashi, M. Morohashi, H. Kitano, Cell designer: a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO* **1**(5), 159–162 (2003)
20. A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, H. Kitano, Cell designer 3.5: a versatile modeling tool for biochemical networks. *Proc. IEEE* **96**(8), 1254–1265 (2008)
21. M.A. Gibson, J. Bruck, Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A* **104**(9), 1876–1889 (2000)
22. D.T. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.* **22**(4), 403–434 (1976)
23. D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977)
24. G. Hajnoczky, L.D. Robb-Gaspers, M.B. Seitz, A.P. Thomas, Decoding cytosolic calcium oscillations in the mitochondria. *Cell* **82**(3), 415–424 (1995)
25. G. Hajnoczky, E. Davies, M. Madesh, Calcium signaling and apoptosis. *Biochem. Biophys. Res. Commun.* **304**(3), 445–454 (2003)

26. Y. Han, M. Wind-Rotolo, H.C. Yang, J.D. Siliciano, R.F. Siliciano, Experimental approaches to the study of HIV-1 latency. *Nat. Rev. Microbiol.* **5**(2), 95–106 (2007)
27. D. Hanahan, R.A. Weinberg, The hallmarks of cancer. *Cell* **100**(1), 57–70 (2000)
28. D. Harman, Role of free radicals in aging and disease. *Annals N.Y. Acad. Sci.* **673**, 126–141 (1992)
29. E.L. Haseltine, J.B. Rawlings, Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *J. Chem. Phys.* **117**(15), 6959–6969 (2002)
30. F. Hashimoto, N. Oyaizu, V.S. Kalyanaraman, S. Pahwa, Modulation of Bcl-2 protein by CD4 cross-linking: a possible mechanism for lymphocyte apoptosis in human immunodeficiency virus infection and for rescue of apoptosis by interleukin-2. *Blood* **90**(2), 745–753 (1997)
31. A.V. Hill, The possible effects of the aggregation of the molecules of hemoglobin on its dissociation curves. *J. Physiol.* **40**, 4–7 (1910)
32. J. Jack, F.J. Romero-Campero, M.J. Perez-Jimenez, O.H. Ibarra, A. Păun, in *Simulating Apoptosis Using Discrete Methods: A Membrane System and a Stochastic Approach*, eds by M. Domaratzki, K. Salomaa. International Conference of Unconventional Computation (Kingston, ON, CA, 2007), pp. 50–63
33. M. Hucka et al., The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **9**(4), 524–531 (2003)
34. J. Jack, F.J. Romero-Campero, M.J. Perez-Jimenez, O.H. Ibarra, A. Păun, Simulating Apoptosis Using Discrete Methods: A Membrane System and a Stochastic Approach, in *International Conference of Unconventional Computation*, ed. by M. Domaratzki, K. Salomaa (Kingston, ON, CA, 2007), pp. 50–63
35. J. Jack, A. Rodriguez-Paton, O.H. Ibarra, A. Păun, Discrete nondeterministic modeling of the FAS pathway. *Int. J. Found. Comput. Sci.* **15**(5), 1147–1162 (2008)
36. J. Jack, A. Păun, A. Rodriguez-Paton, in *Effects of HIV-1 Proteins on the Fas-mediated Apoptotic Signaling Cascade: A Computational Study of T cell Latency*, eds. by D. Corne, P. Frisco, Gh. Paun, G. Rozenberg, A. Salomaa. 10th Workshop on Membrane Computing WMC 2009, vol 5391 (LNCS, 2009), pp. 227–246
37. J. Jack, A. Păun, Discrete modeling of biochemical signaling with memory enhancement. *LNBI Trans. Comput. Syst. Biol.* **5750**, 200–215 (2009)
38. E. Jacotet et al., The HIV-1 viral protein R induces apoptosis via a direct effect on the mitochondrial permeability transition pore. *J. Exp. Med.* **191**(1), 33–45 (2000)
39. S. Ju, D.J. Panka, H. Cui, R. Ettinger, M. El-Khatib, D.H. Sherr, B.Z. Stanger, A. Marshak-Rothstein, Fas(CD95)/FasL interactions required for programmed cell death after T-cell activation. *Nature* **373**(6513), 444–448 (1995)
40. J. Karn, Tackling Tat. *J. Mol. Biol.* **293**(2), 235–254 (1999)
41. J.F. Kerr, A.H. Wyllie, A.R. Currie, Apoptosis: a basic biological phenomenon with wide-ranging implications in tissue kinetics. *Br. J. Cancer* **26**(4), 239–257 (1972)
42. S. Kim, R. Byrn, J. Groopman, D. Baltimore, Temporal aspects of DNA and RNA synthesis during human immunodeficiency virus infection: evidence for differential gene expression. *J. Virol.* **63**(9), 3708–3713 (1989)
43. S. Kinoshita, L. Su, M. Amano, L.A. Timmerman, H. Kaneshima, G.P. Nolan, The T cell activation factor NF-ATc positively regulates HIV-1 replication and gene expression in T cells. *Immunity* **6**(3), 235–244 (1997)
44. F.C. Kischkel, S. Hellbardt, I. Behrmann, M. Germer, M. Pawlita, P.H. Krammer, M.E. Peter, Cytotoxicity-dependent APO-1 (Fas/CD95)-associated proteins form a death-inducing signaling complex (DISC) with the receptor. *EMBO J.* **14**(22), 5579–5588 (1995)
45. H.A. Kramers, Brownian motion in a field of force and the diffusion model of chemical reactions. *Physica* **7**(4), 284–304 (1940)
46. G. Lahav, N. Rosenfeld, A. Sigal, N. Geva-Zatorsky, A.J. Levine, M.B. Elowitz, U. Alon, Dynamics of the p53-Mdm2 feedback loop in individual cells. *Nat. Genet.* **36**(2), 147–150 (2004)
47. K.G. Lassen, K.X. Ramyar, J.R. Bailey, Y. Zhou, R.F. Siliciano, Nuclear retention of multiply spliced HIV-1 RNA in resting CD4+ T cells. *PLoS Pathog.* **2**(7), 650–661 (2006)

48. I.N. Lavrik, A. Golks, P.H. Krammer, Caspases: pharmacological manipulation of cell death. *J. Clin. Inv.* **115**(10), 2665–2672 (2005)
49. V. Manca, The metabolic algorithm for p systems: principles and applications. *Theor. Comput. Sci.* **404**, 142–155 (2008)
50. E. Maniati, P. Potter, N.J. Rogers, B.J. Morley, Control of apoptosis in autoimmunity. *J. Pathol.* **214**(2), 190–198 (2008)
51. D.A. McQuarrie, Stochastic approach to chemical kinetics. *J. Appl. Probab.* **4**(3), 413–478 (1967)
52. L. Montagnier et al., A new type of retrovirus isolated from patients presenting with lymphadenopathy and acquired immune deficiency syndrome: structural and antigenic relatedness with equine infectious anemia virus. *Annales de Viro.* **135**(1), 119–134 (1984)
53. E.W. Montroll, K.E. Shuler, The application of the theory of stochastic processes to chemical kinetics. *Adv. Chem. Phys.* **1**, 361–399 (1958)
54. G. Nabel, D. Baltimore, An inducible transcription factor activates expression of human immunodeficiency virus in T cells. *Nature* **344**(6262), 711–713 (1987)
55. G. Napolitano, P. Licciardo, R. Carbone, B. Majello, L. Lania, CDK9 has the intrinsic property to shuttle between nucleus and cytoplasm, and enhanced expression of cyclinT1 promotes its nuclear localization. *J. Cell. Physiol.* **192**(2), 209–215 (2002)
56. D.E. Nicholson, From bench to clinic with apoptosis-based therapeutic agents. *Nature* **407**(6805), 810–816 (2000)
57. Z. Nie, B.N. Phenix, J.J. Lum, A. Alam, D.H. Lynch, B. Beckett, P.H. Krammer, R.P. Sekaly, A.D. Badley, HIV-1 protease processes procaspase 8 to cause mitochondrial release of cytochrome c, caspase cleavage and nuclear fragmentation. *Cell Death Differ.* **9**(11), 1172–1184 (2002)
58. D.W. Nijhawan, N. Honarpour, X. Wang, Apoptosis in neural development and disease. *Ann. Rev. Neurosci.* **23**, 73–87 (2000)
59. J. O’Connell, M.W. Bennett, G.C. O’Sullivan, J.K. Collins, F. Shanahan, Fas counter-attack - the best form of tumor defense? *Nat. Med.* **5**(3), 267–268 (1999)
60. J. O’Connell, M.W. Bennett, G.C. O’Sullivan, J.K. Collins, F. Shanahan, The Fas counterattack: cancer as a site of immune privilege. *Immunol. Today* **20**(1), 46–52 (1999)
61. K. Oda, Y. Matsuoka, A. Funahashi, H. Kitano, A comprehensive pathway map of epidermal growth factor receptor signaling. *Mol. Syst. Biol.* **1**, 1–17 (2005)
62. Z.N. Oltvai, C.L. Millman, S.J. Korsmeyer, Bcl-2 heterodimerizes in vivo with a conserved homolog, Bax, that accelerates programmed cell death. *Cell* **74**(4), 609–619 (1993)
63. J.T. Opferman, S.J. Korsmeyer, Apoptosis in the development and maintenance of the immune system. *Nat. Immunol.* **4**(5), 410–415 (2003)
64. N. Oyaizu, T.W. McCloskey, S. Than, R. Hu, V.S. Kalyanaraman, S. Pahwa, Cross-linking of CD4 molecules upregulates fas antigen expression in lymphocytes by inducing interferon- $\gamma$  and tumor necrosis factor- $\alpha$  secretion. *Blood* **84**(8), 2622–2631 (1994)
65. T. Ozawa, Mechanism of somatic mitochondrial DNA mutations associated with age and diseases. *Biochim Biophys Acta* **1271**(1), 177–189 (1995)
66. A.S. Perelson, P. Essunger, Y. Cao, M. Vesanen, A. Hurley, K. Saksela, M. Markowitz, D.D. Ho, Decay characteristics of HIV-1-infected compartments during combination therapy. *Nature* **387**(6629), 188–191 (1997)
67. T.C. Pierson, Y. Zhou, T.L. Kieffer, C.T. Ruff, C. Buck, R.F. Siliciano, Molecular characterization of preintegration latency in human immunodeficiency virus type 1 infection. *J. Virol.* **76**(17), 8518–8531 (2002)
68. P. Pinton, M. Brini, C. Bastianutto, R.A. Tuft, T. Pozzan, R. Rizzuto, New light on mitochondrial calcium. *Biofactors* **8**(3), 243–253 (1998)
69. P. Pinton, D. Ferrari, P. Magalhaes, K. Schulze-Osthoff, F. Di Virgilio, T. Pozzan, R. Rizzuto, Reduced loading of intracellular  $\text{Ca}^{2+}$  stores and downregulation of capacitative  $\text{Ca}^{2+}$  influx in Bcl-2-overexpressing cells. *J. Cell. Biol.* **148**(5), 857–862 (2000)
70. P. Pinton, C. Giorgi, R. Siviero, E. Zecchini, R. Rizzuto, Calcium and apoptosis: ER-mitochondria  $\text{Ca}^{2+}$  transfer in the control of apoptosis. *Oncogene* **27**(50), 6407–6418 (2008)

71. C.V. Rao, A.P. Arkin, Stochastic chemical kinetics and the quasi-steady-state assumption: applicaiton to the Gillespie algorithm. *J. Chem. Phys.* **118**(11), 4999–5010 (2003)
72. A.G. Renehan, C. Booth, C.S. Potten, What is apoptosis, and why is it important? *Br. Med. J.* **322**, 1536–1538 (2001)
73. R. Rizzuto, P. Pinton, W. Carrington, F.S. Fay, K.E. Fogarty, L.M. Lifshitz, R.A. Tuft, T. Pozzan, Close contacts with the endoplasmic reticulum as determinants of mitochondrial  $\text{Ca}^{2+}$  responses. *Science* **280**(5370), 1763–1766 (1998)
74. S.L. Rowland-Jones, H.C. Whittle, Out of Africa: what can we learn from HIV-2 about protective immunity to HIV-1. *Nat. Immunol.* **9**(4), 329–331 (2007)
75. C. Scaffidi, S. Fulda, A. Srinivasan, C. Friesen, F. Li, K.J. Tomaselli, K.M. Debatin, P.H. Krammer, M.E. Peter, Two CD95 (APO-1/Fas) signaling pathways. *EMBO J.* **17**(6), 1675–1687 (1998)
76. N. Selliah, T.H. Finkel, Biochemical mechanisms of HIV induced T cell apoptosis. *Cell Death Differ.* **8**(2), 127–136 (2001)
77. J.D. Siliciano et al., Long-term follow-up studies confirm the stability of the latent reservoir for HIV-1 in resting CD4+ T cells. *Nat. Med.* **9**(6), 727–728 (2003)
78. V. Simon, D. Ho, Q. Abdool Karim, HIV/AIDS epidemiology, pathogenesis, prevention, and treatment. *Lancet* **368**(9534), 489–504 (2006)
79. S. Sperandio, I. de Belle, D.E. Bredesen, An alternative, nonapoptotic form of programmed cell death. *Proc. Natl. Acad. Sci.* **97**(26), 14376–14381 (2000)
80. M.C. Strain et al., Heterogeneous clearance rates of long-lived lymphocytes infected with HIV: Intrinsic stability predicts lifelong persistence. *Proc. Natl. Acad. Sci.* **100**(8), 4819–4824 (2003)
81. P.R. Strack et al., Apoptosis mediated by HIV protease is preceded by cleavage of Bcl-2. *Proc. Natl. Acad. Sci.* **93**(18), 9571–9576 (1996)
82. F. Teranishi, Z.Q. Liu, M. Kunimatsu, K. Imai, H. Takeyama, T. Manabe, M. Sasaki, T. Okamoto, Calpain is involved in the HIV replication from the latently infected OM10.1 cells. *Biochem. Biophys. Res. Commun.* **303**(3), 940–946 (2003)
83. J.M.G. Vilar, H.Y. Kueh, N. Barkai, S. Leibler, Mechanisms of noise-resistance in general oscillations. *Proc. Natl. Acad. Sci. USA* **99**(9), 5988–5992 (2002)
84. G.K. Voeltz, M.M. Rolls, T.A. Rapoport, Structural organization of the endoplasmic reticulum. *EMBO R.* **3**(10), 944–950 (2002)
85. K. Wang, X.M. Yin, D.T. Chao, C.L. Millman, S.J. Korsmeyer, BID: a novel BH3 domain-only death agonist. *Genes Deve.* **10**(22), 2859–2869 (1996)
86. L.S. Weinberger, J.C. Burnett, J.E. Toettcher, A.P. Arkin, D.V. Schaffer, Stochastic gene expression in a lentiviral positive-feedback loop: HIV-1 tat fluctuations drive phenotypic diversity. *Cell* **122**(2), 169–182 (2005)
87. L. Wilhelmy, Ueber das Gesetz, nach welchem die Einwirkung der Säuren auf den Rohrzucker stattfindet. *Poggendorff's Annalen der Physik und Chemie* **81**, 413–433 (1850)
88. R. Wyatt, J. Sodroski, The HIV-1 envelope glycoproteins: fusogens, antigens, and immunogens. *Science* **280**(5371), 1884–1888 (1998)
89. Y. Yang, B. Dong, P.R. Mittelstadt, H. Xiao, J.D. Ashwell, HIV Tat binds Egr proteins and enhances Egr-dependent transactivation of the Fas ligand promoter. *J. Biol. Chem.* **277**(22), 19482–19487 (2002)
90. W.R. Yu, T. Liu, T.K. Fehlings, M.G. Fehlings, Involvement of mitochondrial signaling pathways in the mechanism of Fas-mediated apoptosis after spinal cord injury. *Eur. J. Neurosci.* **29**(1), 114–131 (2009)
91. J.A. Zack, S.J. Arrigo, S.R. Weitsman, A.S. Go, A. Haislip, I.S. Chen, HIV-1 entry into quiescent primary lymphocytes: molecular analysis reveals a labile, latent viral structure. *Cell* **61**(2), 213–222 (1990)

92. J.A. Zack, A.M. Haislip, P. Krogstad, I.S. Chen, Incompletely reverse-transcribed human immunodeficiency virus type 1 genomes in quiescent cells can function as intermediates in the retroviral life cycle. *J. Virol.* **66**(3), 1717–1725 (1992)
93. G. Zauli, D. Gibellini, P. Secchiero, H. Dutartre, D. Olive, S. Capitani, Y. Collette, human immunodeficiency virus type 1 nef protein sensitizes CD4+ T lymphoid cells to apoptosis via functional upregulation of the CD95/CD95 ligand pathway. *Blood* **93**(3), 1000–1010 (1999)
94. Y. Zhou, H. Zhang, J.D. Siliciano, R.F. Siliciano, Kinetics of human immunodeficiency virus type 1 decay following entry into resting CD4+ T Cells. *J. Virol.* **79**(4), 2199–2210 (2005)
95. World Health Organization 2007 AIDS Epidemic Update (2007), <http://www.who.int/hiv/en>

# Chapter 7

## MP Modelling for Systems Biology: Two Case Studies

Luca Marchetti, Vincenzo Manca, Roberto Pagliarini  
and Aliccia Bollig-Fischer

**Abstract** Metabolic P systems (MP systems), based on Păun's P systems, were introduced for modelling metabolic systems by means of suitable multiset rewriting grammars. The initial modelling framework has been widely extended in last years and equipped with a new regression algorithm which derives MP models from the time series of observed dynamics. This has allowed us to dramatically extend the range of possible MP modelling applications from metabolic dynamics to more general kinds of dynamical systems. In this work two applications of MP systems are presented, for discovering the internal regulation logic of two phenomena relevant to systems biology. The first one is a metabolic dynamics related to glucose/insulin interactions during the Intravenous Glucose Tolerance Test. The second one deals with the definition of gene expression networks related to breast cancer under the inhibition of a growth factor.

### 7.1 Introduction

An important problem of systems biology is the mathematical definition of *dynamical systems* explaining observed biological dynamics by taking into account what is already known about the underlying phenomenon [3, 12, 17, 24, 27, 30, 31].

---

L. Marchetti (✉) · V. Manca  
Department of Computer Science, University of Verona, Strada Le Grazie 15, 37134 Verona,  
Italy  
e-mail: luca.marchetti@univr.it

V. Manca  
e-mail: vincenzo.manca@univr.it

R. Pagliarini  
Telethon Institute of Genetics and Medicine, Naples, Italy  
e-mail: r.pagliarini@tigem.it

A. Bollig-Fischer  
Department of Oncology, Wayne State University and Barbara Ann Karmanos Cancer Institute,  
Detroit, MI, USA  
e-mail: bollig@karmanos.org

The main analysis framework for the most part of biological dynamics remains the theory of ordinary differential equations (ODEs). *Metabolic P systems (MP systems)*, based on Păun's P systems [54], were introduced in [41] for modelling *metabolic systems* by means of suitable multiset rewriting grammars. They are essentially a particular type of finite difference recurrence equations where “fluxes” (see later) play a role analogous to that of derivatives in ODEs. This change of perspective, from a continuous to a discrete approach, provides in many cases computational and modelling advantages. The following discussion and the results of the present chapter intend to argument important cases showing this kind of advantages.

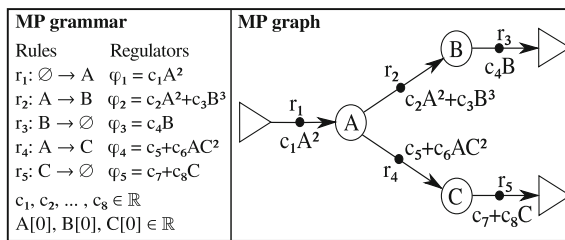
A Metabolic P system is essentially a multiset grammar where multiset transformations are regulated by functions [38, 39]. Namely, a rule like  $a + b \rightarrow c$  means that a number  $u$  of molecules of kind  $a$  and  $u$  of kind  $b$  are replaced by  $u$  molecules of type  $c$ . The value of  $u$  is the *flux* of the rule application. Let us consider a system at some time steps  $i \in \{0, 1, 2, \dots, t\}$ . Let us also assume that a substance  $x$  is produced by rules  $r_1, r_3$  and consumed by rule  $r_2$ . If  $u_1[i], u_2[i], u_3[i]$  are the fluxes of the rules  $r_1, r_2, r_3$  respectively, in the passage from step  $i$  to step  $i + 1$ , then the variation  $\Delta_x[i]$  of substance  $x$  at step  $i$  is given by:

$$x[i + 1] - x[i] = u_1[i] - u_2[i] + u_3[i].$$

In an MP system, in any state, the flux  $u_j$  of rule  $r_j$  is provided by a state function  $\varphi_j$ , called *regulator* of the rule. A state is essentially determined by the quantities of substances defined in the system. However, usually only some substances are arguments of regulators, therefore if  $u_j = \varphi_j(x, y, \dots)$ , the arguments  $x, y, \dots$  of  $\varphi_j$  will be called *tuners* of the regulator.

Substances (also metabolites), rules, initial values and regulators define an *MP grammar*, which is easily representable by an *MP graph* [40]. The set of the rules of an MP grammar can be also represented by a *stoichiometric matrix*  $\mathbb{A}$ , which gives a sort of “matrix-like representation” of the system stoichiometry (see Fig. 7.1). Namely, an MP grammar  $G$  is given by a structure [39]:

$$G = (X, R, \Phi, X_0)$$



**Fig. 7.1** An example of MP grammar (where  $\emptyset$  denotes an empty multiset and substance symbols occurring in regulators denote the corresponding substance quantities). The MP graph on the right is obtained by translating the rules in the source–target–edge notation [40]



where:

1.  $X$  is a finite set of  $n$  substances;
2.  $R$  is a finite set of  $m$  rules and each rule  $r \in R$  is expressed by  $\alpha_r \rightarrow \beta_r$  with  $\alpha_r, \beta_r$  multisets over  $X$  (functions from  $X$  to  $\mathbb{N}$  assigning a *multiplicity* to every substance);
3.  $\Phi = \{\varphi_r \mid r \in R\}$  is the set of *regulators*, or *flux functions*

$$\varphi_r : \mathbb{R}^n \rightarrow \mathbb{R}$$

for every  $r \in R$ , where:

- $\mathbb{R}^n$  is the set of possible *states* of  $X$ ;
- any regulator  $\varphi_r$  associates a flux value  $u_r$  to every state of  $\mathbb{R}^n$ . A flux  $u_r$  of a rule  $r$  establishes an updating of the current state of substances, by decreasing of  $u_r \cdot \alpha_r(x)$  the quantity of any substance  $x$  occurring in  $\alpha_r$  and by increasing of  $u_r \cdot \beta_r(y)$  the quantity of any substance  $y$  occurring in  $\beta_r$  (where  $\alpha_r(x)$  and  $\beta_r(y)$  are the multiplicities of  $x$  and  $y$  in  $\alpha_r$  and in  $\beta_r$ , respectively). This reading of rules leads to the calculation of the following discrete dynamics ( $x[i] \mid i \in \mathbb{N}$ ) for any substance  $x$ , starting from the given initial state  $X_0$  (see the next point), called *Equational Metabolic Algorithm* (EMA):

$$x[i+1] - x[i] = \sum_{j=1}^m (\beta_j(x) - \alpha_j(x)) \cdot u_j[i]. \quad (7.1)$$

4.  $X_0$  is a state of  $X$ , called initial state of  $G$ .

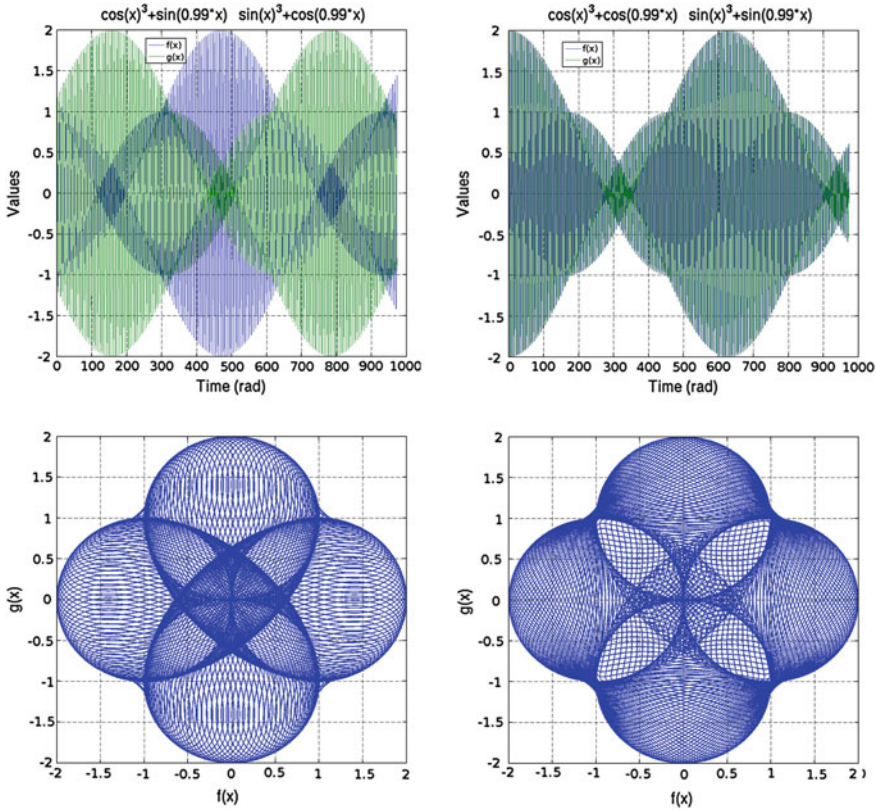
An MP system is essentially an MP grammar equipped with a *temporal interval*  $\tau$ , a conventional *mole size*  $v$ , and substances masses, which specify the time and population (discrete) granularities respectively [38, 39]. In the following, the MP dynamics we will present are computed in MATLAB<sup>1</sup> by applying the EMA formula given in (7.1).

The dynamics which can be modelled by MP systems can be very complicated even by considering simple MP grammars (i.e. with few substances and linear regulators). In [43] MP systems were successfully applied to the field of real periodical function approximation. In that work, we presented some interesting MP oscillators which are obtained by approximating the plot of some given periodical functions.<sup>2</sup> In Fig. 7.2 the dynamics of two MP oscillators are depicted, which are approximated by an MP grammar with only six substances and linear regulators. The complexity of the dynamics compared to the simplicity of the MP grammar which calculates it by EMA, suggests that MP system theory can be a suitable framework for modelling biological dynamics.

In the following, two applications of MP systems will be presented for discovering the internal regulation logic of phenomena relevant to systems biology:

<sup>1</sup> See <http://www.mathworks.it/index.html> for details on the MATLAB software.

<sup>2</sup> The approximation order ranges from  $10^{-6}$  to  $10^{-14}$ , depending on the considered model.



**Fig. 7.2** Examples of complicated oscillators which can be obtained with simple MP grammars with linear regulators (see [43] for details)

1. the glucose/insulin dynamics in the Intravenous Glucose Tolerance Test (IVGTT);
2. the modelling of gene expression networks.

Despite the differences between the considered phenomena, in both the cases a model was found that exhibits good approximation of the observed time series and highlights results which are new or that have been only theorized (see also [47, 48]).

### 7.1.1 Log–Gain Stoichiometric Stepwise Regression (LGSS)

The results obtained in [43] suggested some possible applications to specific cases of interest. In particular, the procedure introduced to define the models has been widely extended in [44–46] for defining *LGSS (Log–Gain Stoichiometric Stepwise Regression)*, a regression algorithm which derives MP models from the time series of observed dynamics. All the MP models given in the following have been defined by means of this algorithm.

LGSS can be applied independently from any knowledge about reaction rate kinetics and it represents the most recent solution, in terms of MP systems, of the *dynamical inverse problem*, that is, of the identification of (discrete) mathematical models exhibiting an observed dynamics and satisfying all the constraints required by the specific knowledge about the modelled phenomenon. The LGSS algorithm combines and extends the log–gain principles developed in the MP system theory [36, 37] with the classical method of Stepwise Regression [25], which is a statistical regression technique based on Least Squares Approximation and statistical F–tests [16].

LGSS has been implemented in 2010 as a set of MATLAB functions. All the functions have been ad hoc implemented and do not require additional toolboxes.<sup>3</sup> The most computationally demanding code (regression, simulation and tuning of regression parameters) has been implemented by taking advantage of the parallel processing facilities offered by the Parallel Computing Toolbox (the software, however, runs also when this toolbox is not installed). The implementation considers also some extensions of the algorithm which permit to add constraints. In particular, when the Optimization Toolbox is installed in the system, LGSS supports also the usage of the *lsqlin* function which computes *constrained linear least squares problems*. This last feature is very important when it is needed to force complex constraints on the least squares estimation of the computed regressor coefficients.

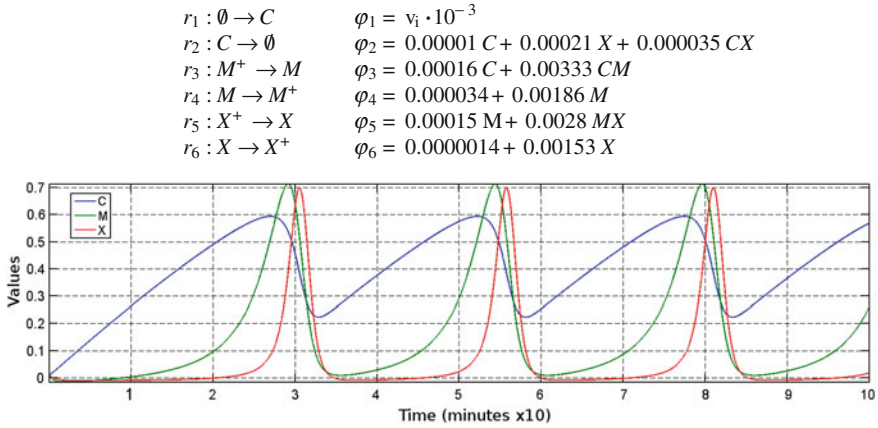
LGSS is highly customizable: all the features, all the thresholds used by the code can be configured initially or at runtime. In order to start, LGSS requires the stoichiometry of the MP system (i.e. the set of the rules), the time series of substances, and, finally, the set of basic functions that the user wants to consider in representing regulators as linear combination of them (we call this set the *regressor dictionary*).

Even if computational tools are available for evaluating unknown parameters of ODE models [26, 34, 52, 59], LGSS seems to point out a general methodology for solving dynamical inverse problems. In fact, LGSS not only discovers unknown parameters, but suggests also the form of regulators as a combination of basic functions among those specified by the user in the regression dictionary. This possibility could be very important in the case where the knowledge about the phenomenon under investigation is so poor that no clear idea is available about the kind of model underlying the observed behaviour.

The size of the systems of equations solved by LGSS depends on the number of substances and reactions of the MP system under examination and on its temporal interval  $\tau$  (a smaller temporal interval requires longer time series and so a larger system of equations). However, the regression usually ends in few minutes (less than one minute in many cases, using a common laptop with a dual core CPU and 4 Gbyte of RAM memory), but it can increase to hours when the system is very big (i.e. a system with several thousands of equations, and a regression dictionary of hundreds of regressors).

---

<sup>3</sup> Therefore, LGSS can be executed by other “MATLAB–like” free applications (for example, the GNU Octave project at <http://www.octave.org>).

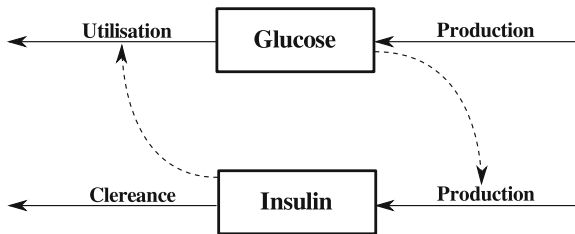


**Fig. 7.3** Example of MP mitotic oscillator with  $\tau = 10^{-3}$  min

In [42] LGSS has been applied to Goldbeter's oscillator<sup>4</sup> for showing that MP systems yield a robust method for biological modelling. In this manner, were automatically generated 700 models of this oscillator, which, for the most part, provide the same order of approximation of Goldbeter's model for different values of  $\tau$  (see Fig. 7.3 for an example). In that case, LGSS was able to complete the regression of all the 700 models in few hours.

## 7.2 The Glucose/Insulin Dynamics in the Intravenous Glucose Tolerance Test (IVGTT)

*Glucose* is the primary source of energy for body cells. It is transported from the intestines or liver to body cells via the bloodstream, and is absorbed by the cells with



**Fig. 7.4** Schematic diagram of the glucose–insulin regulatory system

<sup>4</sup> The Goldbeter's mitotic oscillator represents the simplest form of mitotic mechanism found in early amphibian embryos [22].

the intervention of the hormone *insulin* produced by the pancreas (see Fig. 7.4). Blood glucose concentration is a function of the rate of glucose which enters the bloodstream, the glucose appearance, balanced by the rate of glucose which is removed from the circulation, the glucose disappearance. Normally, in mammals this concentration is tightly regulated as a part of metabolic homeostasis.

If the plasma glucose concentration level is constantly out of the usual range, then we are in presence of blood glucose problems. In particular, when this level is constantly higher than the range upper bound (which is referred to as *hyperglycemia*), we are in presence of *Diabetes*: a dreadfully severe and pervasive illness which concerns a good number of organs in the body. Diabetes is classified into two main categories known as *type 1* and *type 2*, respectively. In both types of diabetes, the illness can lead to several complications like retinopathy, nephropathy, peripheral neuropathy and blindness. This motivates researches to study the glucose–insulin endocrine regulatory system. In particular, the glucose–insulin system has been the object of repeated, mathematical modelling attempts.

The great majority of the proposed models were devoted to the study of the glucose–insulin dynamics by considering experimental data obtained by the *intravenous glucose tolerance test*, shortly *IVGTT*, and the *oral glucose tolerance test*, shortly *OGTT*. In these models, the insulin–glucose system is assumed to be composed of two linked subsystems modelling the insulin action and the glucose kinetics, respectively. Since the action of insulin is delayed with respect to plasma glucose, the subsystems of insulin action typically includes a delay. The different strategies which can be used to model such kind of delay will be widely discussed in the following sections.

The *intravenous glucose tolerance test* focuses on the metabolism of glucose in a period of 3 h starting from the infusion of a bolus of glucose at time  $t = 0$  (see Fig. 7.5). IVGTT has been recommended as a method to assess the use of insulin

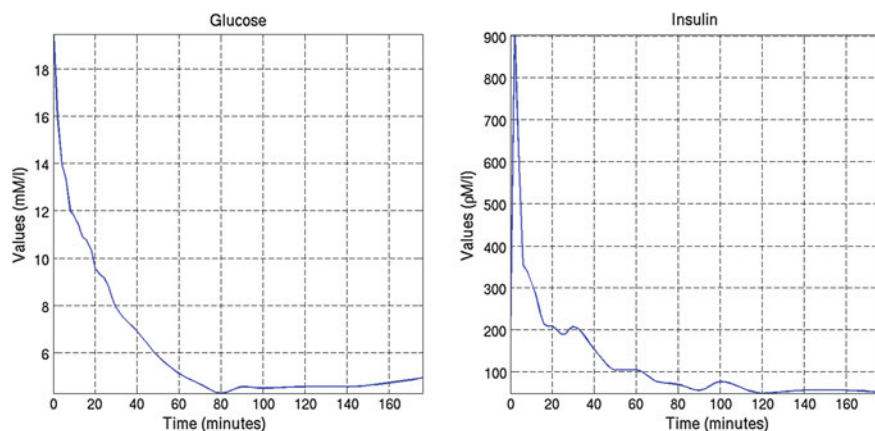


Fig. 7.5 Plots of a IVGTT data-set starting from the time of the glucose injection

in order to identify subjects which may be diabetic [51]. However, considering the limits of the existing mathematical models (see the next section), a need exists to have reliable mathematical models representing the glucose–insulin system. The mere fact that several models have been proposed [9, 35, 49] shows that mathematical and physiological considerations have to be carefully integrated when attempting to represent the glucose–insulin regulatory mechanism. In particular, in order to model the IVGTT, a reasonably simple model is required. It has to have a few parameters to be estimated and its dynamics has to be consistent with physiology and with experimental data. Further, the model formulation, while applicable to model the IVGTT, should be logically and easily extensible to model other envisaged experimental procedures.

### 7.2.1 Mathematical Models of the Intravenous Glucose Tolerance Test

A variety of mathematical models, statistical methods and algorithms have been proposed to understand different aspects of diabetes. In this section we briefly recall the two mathematical models which had the most important impact in diabetology for modelling the IVGTT by considering only the dynamics of glucose and insulin.<sup>5</sup>

Although several other models have been proposed [4], the real start of modelling glucose–insulin dynamics is due to the *minimal model* developed in [5, 61]. It is based on the following system of differential equations:

$$\begin{aligned}\frac{dG(t)}{dt} &= -(p_1 + X(t)) G(t) + p_1 G_b \\ \frac{dX(t)}{dt} &= -p_2 X(t) + p_3 (I(t) - I_b) \\ \frac{dI(t)}{dt} &= p_4 (G(t) - p_5) - p_6 (I(t) - I_b)\end{aligned}\tag{7.2}$$

where  $G(t)$  [ $mg/dl$ ] and  $I(t)$  [ $\mu UI/ml$ ] are plasma glucose and insulin concentrations at time  $t$  [ $min$ ], respectively, and  $(G(t) - p_5)$  is assumed to be 0 when  $G(t) < p_5$ . The auxiliary function  $X(t)$  [ $min^{-1}$ ] models the time delay of the insulin consumption on glucose.  $G_b$  and  $I_b$  are the subject baseline blood glucose and insulin concentration, while  $p_i$ , for  $i = 1, 2, \dots, 6$ , are the model parameters (we refer the reader to [5, 61] for all the details concerning these parameters).

Although (7.2) is very useful in physiology, it is based on some oversimplified mathematical representations. In fact, the artificial non-observable variable  $X(t)$  is introduced to model the delay in the action of insulin, but it has no precise biological

---

<sup>5</sup> Other models currently available in literature provide a more accurate modelling of the test by considering other factors also, such as the dynamics of C-peptide, which is secreted by the pancreas at the same rate of insulin (see [62] for details).

meaning. Therefore the *dynamical model* has been proposed in [18]:

$$\begin{aligned}\frac{dG(t)}{dt} &= -b_1 G(t) - b_4 I(t) G(t) + b_7 \\ G(t) &\equiv G_b \quad \forall t \in [-b_5, 0) \\ \frac{dI(t)}{dt} &= -b_2 I(t) + \frac{b_6}{b_5} \int_{t-b_5}^t G(s) ds.\end{aligned}\tag{7.3}$$

It is a delay integro–differential equation model which is a more realistic representation of the glucose–insulin dynamics which follows an IVGTT. The term  $\frac{b_6}{b_5} \int_{t-b_5}^t G(s) ds$  represents the *decaying memory kernel* [15], which is introduced to model the time delay. The physiologic meaning of the delay kernel reflects the pancreas sensitivity to the blood glucose concentration in the past.

An extension of (7.3) is proposed in [50], where a generic weight function  $\omega$  is introduced in the delay integral kernel modelling the pancreatic response to glucose level. In this way, the second equation of (7.3) becomes:

$$\frac{dI(t)}{dt} = -b_2 I(t) + b_6 \int_0^\infty \omega(s) G(t-s) ds \tag{7.4}$$

where  $\omega(s)$  is assumed to be a non–negative square integrable function on  $\mathbb{R}^+ = [0, \infty)$ , such that  $\int_0^\infty \omega(s) ds = 1$  and  $\int_0^\infty s \cdot \omega(s) ds$  is equal to the average time delay. The idea is that different patient populations show different shapes of the kernel function  $\omega$ , and then suitable parametrization of this function could offer the possibility to classify patients by means of experimental parameter identification.

Despite the models (7.3) and (7.4) solve the drawbacks of the minimal model, they made some assumptions that may not be realistic. The main restriction regards the way used to introduce the delay on the basis of subjective assumptions. This suggests the study of other ways to model the IVGTT process.

## 7.2.2 MP Modelling of IVGTT

In the previous section we introduced some ODE systems that model the IVGTT under different assumptions. The main difference between the models is the way adopted for modelling the time delays which occur during the pancreatic insulin secretion. From a mathematical point of view, the problem of coping with delays in ODE systems is difficult because delays are better managed with discrete models. In fact, the perspective of continuous time makes hard to manage finite time intervals which are at the basis of the modelling of time delays. For this reason, beside the theory of Delay differential equations [6], new hybrid approaches are emerging which try to incorporate discrete time series into classical differential models [63].

In this section we try to solve the problem in a complete discrete way, by applying MP systems and LGSS for modelling IVGTT (see [47] for initial results). In order to cope with time delays, we assume that the flux which models the pancreatic



insulin secretion can depend on the glucose concentration at the current time and at some time-steps in the past (glucose memories). This suggests more natural and detailed mechanisms of delays which act in the insulin production. If we indicate by  $G^{(t)} = (G[i] | 0 \leq i \leq t)$  the vector containing the time-series of glucose in a given data-set, we define the time-series  $G_{-m}^{(t)}$  related to the memory of glucose shifted  $m$  steps after as the vector

$$G_{-m}^{(t)} = (\underbrace{G_b, G_b, \dots, G_b}_{m \text{ times}}, G[0], G[1], \dots, G[t-m]) \quad (7.5)$$

where  $G_b$  is the basal value of the blood glucose level.<sup>6</sup> Memories are very simple to be managed in MP systems and increase a lot the approximation power of the models as showed in [43], where memories have been applied in the context of periodical function approximation.

In our analysis we considered ten different data-sets obtained by applying the intravenous glucose tolerance test to ten healthy patients [18, 50, 53]. All subjects have negative family histories for diabetes and other endocrine diseases. During the test, the patients did not get medications and had no current illness. Each test has been performed during the morning after an overnight fast, and for the three days preceding the test each subject followed a diet composed of 55 % carbohydrates, 30 % fats, and 15 % proteins. The curves of the considered data-sets are very different from each other, especially the curve related to the insulin dynamics which exhibits values and peaks of different height and at different delays. In all the cases, however, we found MP models which provide good data fitting (the average of the calculated multiple coefficients of determination [2] for all the models is greater than 0.95 for both glucose and insulin).

In Fig. 7.6 we provide the MP grammars related to four of the considered data-sets, and the plotting of the corresponding calculated dynamics for the insulin. The depicted dynamics exhibit examples of all the different scenarios we observed concerning the insulin release in our data-sets. There can be situations where the insulin curve exhibits many peaks which model the different release phases, or there can be dynamics without significant peaks but that are in any case modelled by a delayed insulin secretion (this is the case of data-set 1). Each MP grammar is given by 2 substances: (i)  $G$ , which represents the blood glucose level, and (ii)  $I$ , which represents the level of insulin, and 4 rules. The first two rules are related to glucose and the others related to insulin: (i)  $r_1$ : constant release of glucose in the blood, (ii)  $r_2$ : glucose disappearance due to a term which represents the normal decay of glucose (depending on  $G$ ) and to a term which indicate the action of insulin (depending on both  $G$  and  $I$ ), (iii)  $r_3$ : delayed release of insulin by the pancreas which depends on the blood glucose level (with memories), and (iv)  $r_4$ : normal decay of insulin.

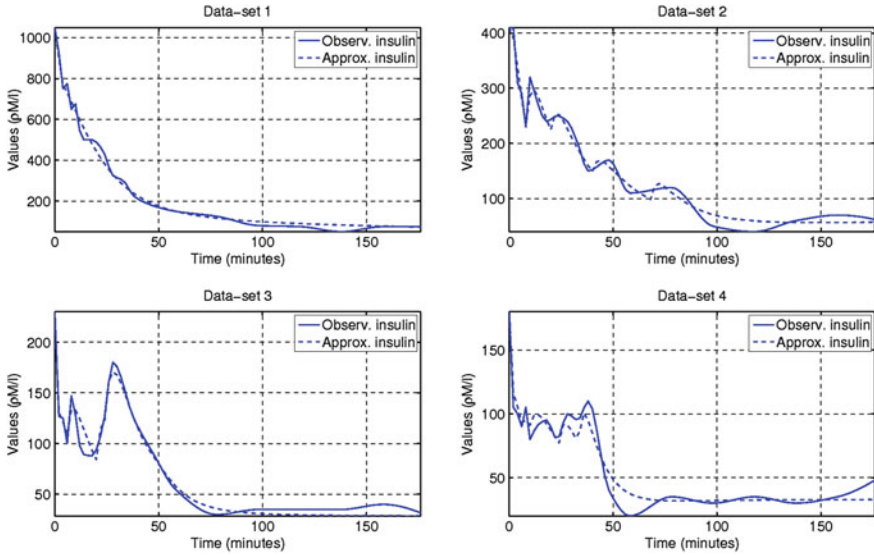
The total number of regressors selected for defining the pancreatic insulin secretion (regulator  $\varphi_3$ ) can be changed by acting on the thresholds used by

---

<sup>6</sup> Since during the IVGTT the glucose level gradually returns to its basal level, here we assume  $G_b$  to be equal to the last value of the considered glucose time-series.

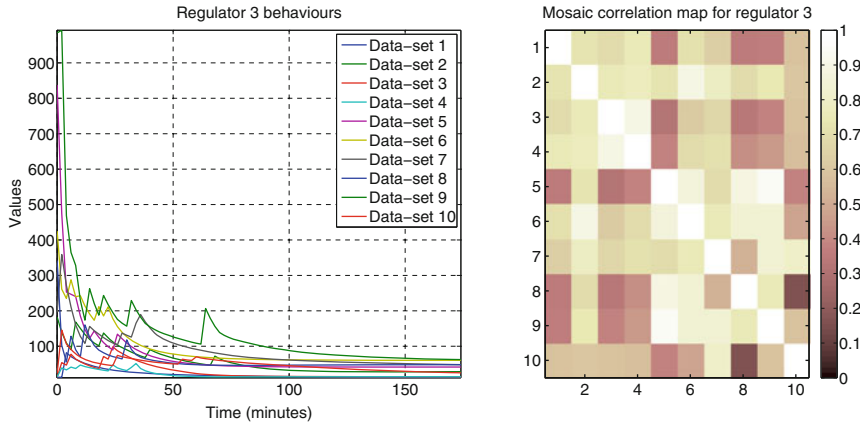


1	$r_1 : \emptyset \rightarrow G$ $r_2 : G \rightarrow / 0$ $r_3 : \emptyset \rightarrow I$ $r_4 : I \rightarrow / 0$ Initial values:	$\phi_1 = 0.011$ $\phi_2 = 6.6 \cdot 10^{-5} GI$ $\phi_3 = 0.5G_{-4}^2$ $\phi_4 = 0.16I$ $I[0] = 1050.0, G[0] = 12.8$
2	$r_1 : \emptyset \rightarrow G$ $r_2 : G \rightarrow / 0$ $r_3 : \emptyset \rightarrow I$ $r_4 : I \rightarrow / 0$ Initial values:	$\phi_1 = 0.056$ $\phi_2 = 5.2 \cdot 10^{-4} I + 8.1 \cdot 10^{-5} GI$ $\phi_3 = 3.76 \cdot 10^{-6} G^7 + 0.74G_{-8}^2 + 0.02G_{-20}^3 + 0.21G_{-40}^2 + 10^{-4}G_{-68}^5$ $\phi_4 = 0.49I$ $I[0] = 410.0, G[0] = 12.2$
3	$r_1 : \emptyset \rightarrow G$ $r_2 : G \rightarrow / 0$ $r_3 : \emptyset \rightarrow I$ $r_4 : I \rightarrow / 0$ Initial values:	$\phi_1 = 0.12$ $\phi_2 = 0.02G + 1.9 \cdot 10^{-4} GI$ $\phi_3 = 0.04G_{-2}^3 + 3.3 \cdot 10^{-5} G_{-6}^6 + 0.44G_{-20}^2 + 0.04G_{-24}^3$ $\phi_4 = 0.5I$ $I[0] = 230.0, G[0] = 10.1$
4	$r_1 : \emptyset \rightarrow G$ $r_2 : G \rightarrow / 0$ $r_3 : \emptyset \rightarrow I$ $r_4 : I \rightarrow / 0$ Initial values:	$\phi_1 = 0.11$ $\phi_2 = 6.2 \cdot 10^{-4} GI$ $\phi_3 = 0.1G_{-2}^2 + 0.9G_{-6} + 1.07G_{-10} + 2.4 \cdot 10^{-4} G_{-24}^4$ $+ 5.4 \cdot 10^{-7} G_{-32}^6 + 5.3 \cdot 10^{-8} G_{-34}^7$ $\phi_4 = 0.4I$ $I[0] = 180.0, G[0] = 16.7$



**Fig. 7.6** MP grammars and calculated insulin dynamics related to four of the considered data-sets ( $\tau = 2$  min, see also [47]). In the plots above, *continuous lines* refer to the observed dynamics, while *dashed lines* refer to the dynamics computed by MP grammars. The MP models presented above can be simulated by means of the MetaPlab software [1]

LGSS during the computing of its statistical tests. The models provided here have been defined trying to balance their simplicity with their power of approximation. In fact, LGSS takes advantage of the discrete nature of the MP systems



**Fig. 7.7** Behaviours (*leftmost*) and mosaic correlation map (*rightmost*) of the computed  $\varphi_3$  time series for the ten data-sets

framework for automatically calculating the best form for  $\varphi_3$  as a linear combination of monomials of glucose concentration and of its memories. Each model provides a sort of picture of the metabolism of the subject who has been analysed. This is reflected in the form of the regulators which is different in each model. The form of  $\varphi_3$  changes according to the different pancreatic response to the increasing of the blood glucose level which we found to be different for each person. This confirms experimentally the idea introduced in the analysis of the dynamical model [50] regarding the different forms of the kernel function  $\omega$  in (7.4). In that work, the goal was the analysis of qualitative properties for a family of kernel functions without a suitable parametrization of  $\omega$ . In our work we have a data-driven way to model the delay action of the pancreatic responses. In Fig. 7.7 we report the behaviours and the mosaic correlation map (a color representation of the correlation matrix computed by considering the Pearson's correlation coefficient [56]) of each calculated  $\varphi_3$  time series. By analysing the correlations between the regulators, we observe that some of them are uncorrelated while others exhibit common behaviours. This is emphasized by the dendrogram given in Fig. 7.8 which shows the grouping of the computed MP models according to the correlation distance between their  $\varphi_3$  time series. By considering a threshold on the maximum correlation distance between the regulators, we can cluster the models in different families (4 in Fig. 7.8) which collect patients with common pancreatic response. This is an experimental evidence that patients can be differentiated by considering the behaviour of  $\varphi_3$ , as suggested in [50] where the possibility of differentiating between patient populations is indicated, by considering the form and the parameters of the kernel function.

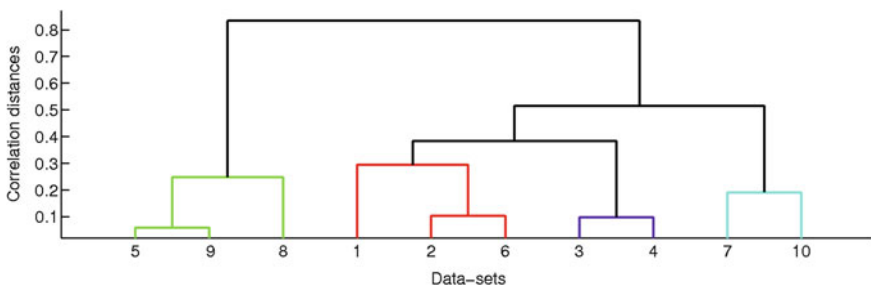
Even if we found differences in the regulation governing the release of insulin, it is possible to recognize a *common pattern* as represented in Fig. 7.9. Here a common logic in the usage of memories becomes evident. Moreover we distinguish two peaks in the first ten minutes which agree with literature. In vivo, insulin secretion is

biphasic with a first phase burst in insulin secretion occurring within the first ten minutes and a second phase that is long some hours [19]. The two peaks we observe perfectly fit with the first phase of insulin secretion and recall the first and the second pancreatic peaks introduced in the analysis of the minimal model [5, 61]. The strength of the peaks is emphasized by the chart in the middle of Fig. 7.9 where the memory usage is weighted with respect to the degree of the corresponding monomials used in  $\varphi_3$ . Here we can see that the first peak is twice the second one and that the release of insulin follows an oscillatory pattern according to experimental results, as reported in [21].

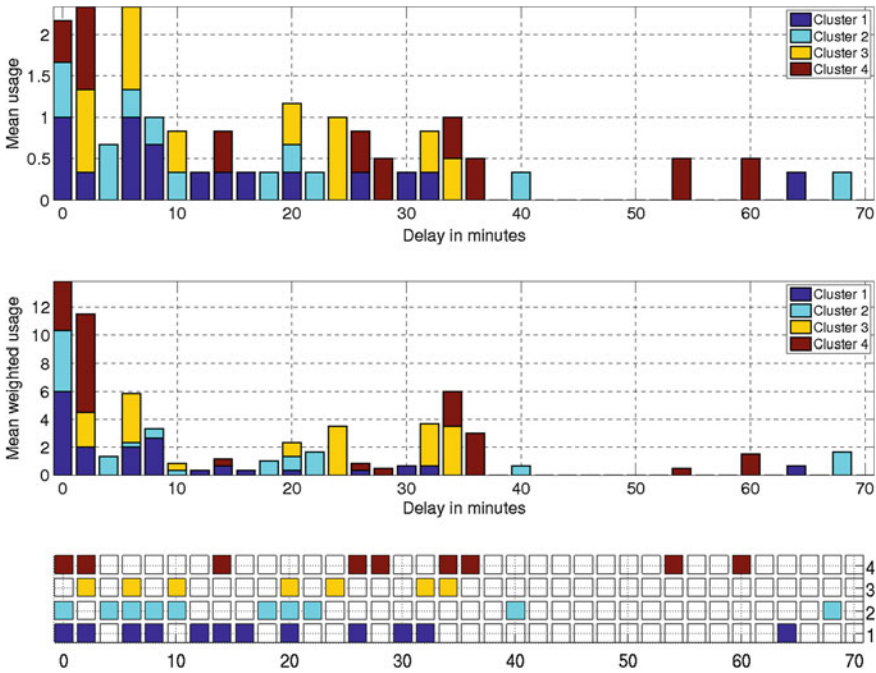
The models in each cluster of Fig. 7.8 provide a release of insulin governed by a specific pattern which contributes to the common behaviour represented in Fig. 7.9. The differences between clusters concern with the delays used during the phases of insulin secretion. In the first phase (first 10 min), three clusters exhibit both the two peaks reported in [5, 61]. However, we can distinguish different delays. This is more evident in the cluster three where also the first peak is delayed (see the chart on the bottom of Fig. 7.9). The fourth cluster does not exhibit two peaks, but this is balanced by a longer and intermittent usage of memories in the first hour (second phase of insulin secretion).

Since the form of  $\varphi_3$  changes, also the form of  $\varphi_2$  (which models the glucose disappearance) is slightly different among models. This is due to the glucose homeostasis: different subjects have different pancreatic response which results in different insulin blood levels. All the non-diabetic subjects, however, need to reach their basal blood glucose level exhibiting very similar glucose time-series. This means that the form of  $\varphi_2$ , which accounts for the glucose disappearance, needs to change its form to cope with the different metabolism of insulin. Despite this difference, the fluxes of  $r_2$  are the same in all the inferred MP models. In fact, the mosaic correlation map depicted in Fig. 7.10 shows high similarities.

We stress here that our regression approach allows us to do a quantitative analysis which makes possible to highlight results which before were only theorized. Further analysis will permit to characterize the differentiation between subjects explained



**Fig. 7.8** Dendrogram showing how the MP models can be grouped by considering the maximum correlation distance between their  $\varphi_3$  time series. The correlation distance is defined as one minus the Pearson's correlation coefficient [56] between each computed flux. The colours divide groups which are distant more than 0.3

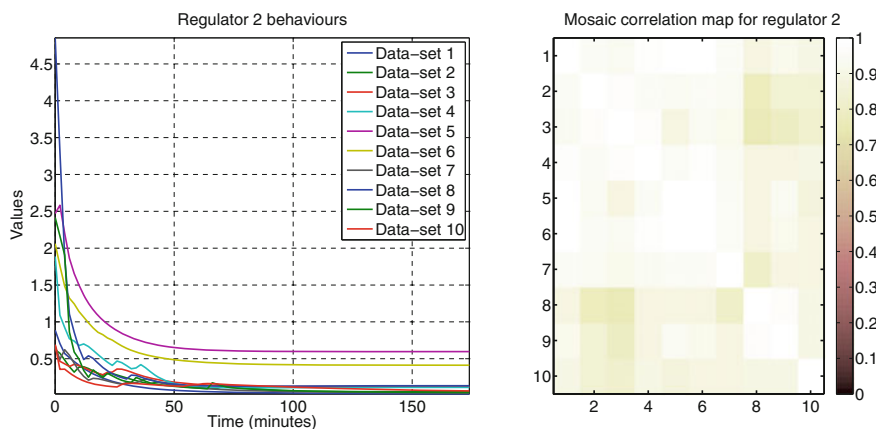


**Fig. 7.9** Bar charts which give the fraction of MP models that use a memory with a given delay (on the x-axis) for each cluster of Fig. 7.8. In the chart on the middle the number of models is weighted by the degree of the monomials used in the models. The chart on the bottom is a 2-D representation of the memory usage for each cluster

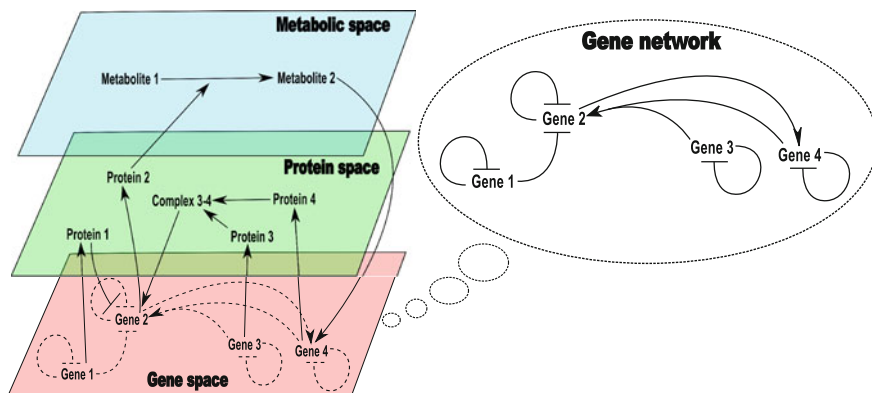
above, by considering physiological parameters such as the height, the weight, the work, the sport activity, and so on. These further analysis may be relevant for defining new reliable mathematical models, that can be used for the development of artificial pancreas [13].

### 7.3 MP Modelling of Gene Networks

As it was shown in previous sections, metabolic phenomena are naturally represented in terms of MP systems. In this section we give an example of how the same modelling platform can be used to model other kinds of biological networks. In fact, by using a suitable translation of MP grammars as gene regulatory networks [48], we can discover, by means of LGSS, their regulatory mechanisms from time series of gene expression levels. Biological networks at different levels are depicted in Fig. 7.11. The way of modelling them by MP grammars are different according to the level that is considered.



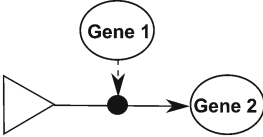

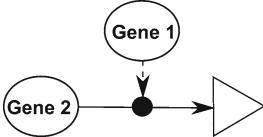

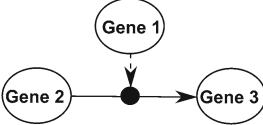
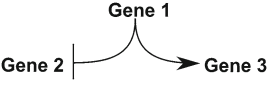
**Fig. 7.10** Behaviours (*leftmost*) and mosaic correlation map (*rightmost*) of the computed  $\phi_2$  time series for the ten data-sets



**Fig. 7.11** An example of a global biochemical network [10]. Molecular constituents (nodes of the network) are organized in three levels (*spaces*): mRNAs, proteins, and metabolites. Solid arcs indicate interactions (*arrows* mean activation, *bars* mean repression). Three different mechanisms of gene-to-gene interactions are shown: (i) regulation of *Gene 2* by the protein product of the *Gene 1*; (ii) regulation of the *Gene 2* by the *Complex 3-4* formed by the products of *Gene 3* and *Gene 4*; and (iii) regulation of *Gene 4* by the *Metabolite 2*, which in turn is produced by *Protein 2*. Projections of these interactions into the “gene space”, indicated by dashed lines, constitute the corresponding gene network (represented also in the right part of the figure)

The identification of new gene networks is now an important part of systems biology [27, 31, 57]. In fact, gene networks might provide valuable clues and new ideas for treating complex diseases and tailoring drug therapy to the individual needs [10, 32]. In addition to high-throughput experimental methods, mathematical and computational approaches are indispensable for the analysis of gene networks. Mathematical modelling supported by computer tools can contribute to the analysis

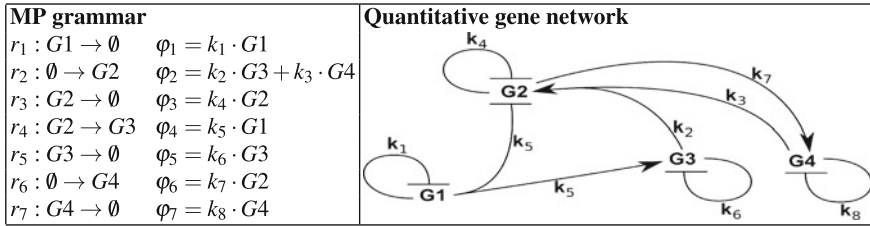
**Table 7.1** Examples of MP reactions and their graphical representations as MP graph and gene network (arrows mean promotion, bars mean inhibition). Please, refer to [48] for details

MP grammar	MP graph	Gene network
<b>Simple promotion</b> $r : \emptyset \rightarrow \text{Gene } 2$ $\varphi : k_1 \cdot \text{Gene } 1$		
<b>Simple inhibition</b> $r : \text{Gene } 2 \rightarrow \emptyset$ $\varphi : k_1 \cdot \text{Gene } 1$		
<b>Simple promotion/inhibition</b> $r : \text{Gene } 2 \rightarrow \text{Gene } 3$ $\varphi : k_1 \cdot \text{Gene } 1$		

of a regulatory network by allowing the biologist to focus on a restricted number of plausible hypotheses. Many reviews of the modelling and simulation of gene networks have been published in recent years (e.g. [7, 11, 20, 23, 29, 60]), presenting the wide variety of formalisms that have been proposed in the literature, such as oriented graphs, Bayesian networks, Boolean networks, differential equations, stochastic master equations and stochastic P systems.

In the following, genes will be modelled as substances whose concentration is equal to the corresponding gene expression level (usually  $\log_2$  transformed, as usual in the context of gene expression analysis). Reactions will model promotion/inhibition between genes which cause the increasing/decreasing in time of the expression levels of the involved genes. The procedure for the definition of the stoichiometry of the system should combine the knowledge about the phenomenon with the analysis of the gene expression profiles (this depends on the problem under investigation). The form of the regulators, instead, is automatically calculated by LGSS (in the same manner we did in the previous section) by obtaining an MP grammar. In [48] we found a standard way for translating an MP grammar involving gene expressions into a corresponding gene network (see Table 7.1 for some examples).

In Fig. 7.12 we provide the MP grammar which is related to the gene network of Fig. 7.11. The stoichiometry of the system and the form of the regulators give the details of the regulations which provide the observed dynamics. The form of the regulator  $\varphi_2$ , for example, indicates that *Gene 2* is promoted by the combined action of *Gene 3* and *Gene 4*. This fact suggests the formation, at the level of the “protein space”, of the *Complex 3–4* represented in Fig. 7.11. The stoichiometry of the rule



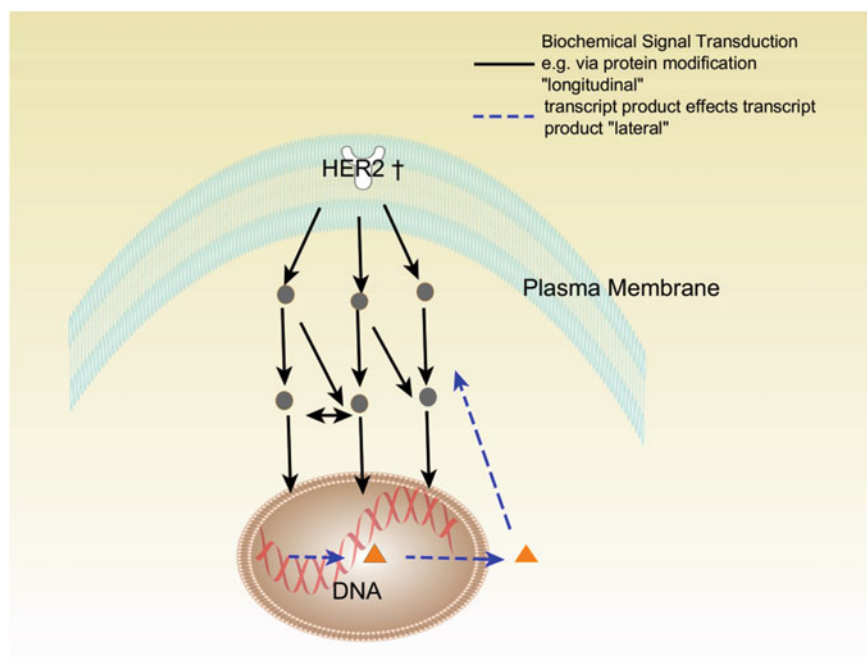
**Fig. 7.12** The MP grammar related to the gene network of Fig. 7.11. The corresponding gene network is reported also in the *right part* of the table enriched with regulator coefficients which provide a measure of the strength of the regulations ( $G1$ ,  $G2$ ,  $G3$  and  $G4$  denote *Gene 1*, *Gene 2*, *Gene 3* and *Gene 4*, respectively)

$r_4$ , instead, indicates the combined action of *Gene 1* which promotes *Gene 3* and, at the same time, inhibits *Gene 2*. Moreover, since the MP regulator coefficients can be considered here as a measure of the strength of the corresponding regulations, we can use these information for enriching the corresponding gene network as displayed in the right part of Fig. 7.12. In conclusion, MP modelling of gene networks not only identifies the genes that affect other genes, but also estimates the strength of such effects by inferring *quantitative gene networks* [32].

### 7.3.1 From Raw Data to MP Models

Genomics and gene expression experiments are very often described as “fishing expeditions” in which the goal is the individuation of new genes involved in a pathway, potential drug targets or expression markers that can be used in a predictive or diagnostic fashion [33]. The usual design for such kind of experiments requires the time series of gene expression profiles for the entire genome of some target cells after having treated them with some specific inhibitors or some targeted up-regulators. The time series are obtained by means of microarray or RNA-Seq analysis [33, 64] providing the same kind of gene expression levels at different time points separated by a given time interval. The raw data will be then processed and analysed in order to get a model which explains the gene regulations which act during the experiment. In our case, the time series are used to run LGSS and obtain the MP model and the corresponding gene network of the phenomenon.

The number of the raw time series data points typically processed for a whole-genome experiment using human cells is usually of the order of tens of thousands. Generally, however, only a small part are really important since many of them refer to genes whose expression profiles exhibit a pattern which is considered to be not related to the phenomenon under examination. For this reason, before we start with the definition of the MP model, raw data need to be preprocessed following a methodology which comprises normalization, filtering and clustering. This methodology,



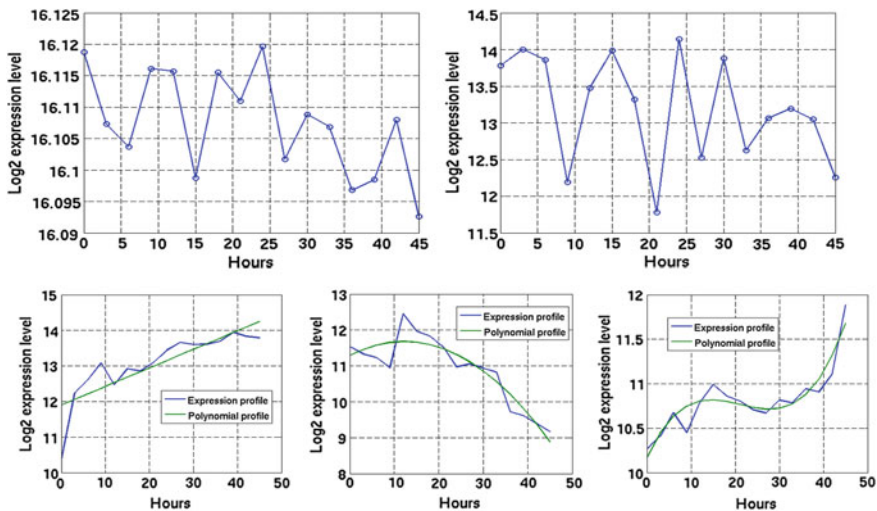
**Fig. 7.13** The action of the HER-2 growth factor on the cell transcriptome

introduced in [48], was successfully applied as a strategy to define the global gene expression network regulated by the HER-2 oncogene.

HER-2 (Human Epidermal Growth Factor Receptor 2) is a receptor tyrosine kinase and a dominant oncogene that is known to drive breast cancer growth. In Fig. 7.13 we remark that the protein-level, biochemical-based network downstream of HER-2 signalling ultimately acts on DNA transcription (continuous arrows). This in turn is joined with the effect of transcript products (dashed arrows), that activate transcription further or feedback on the system. This feedback contributes to non-linear behaviour in gene regulation networks. This phenomenon was analysed in terms of MP regulations and by integrating knowledge of the specific roles of genes highlighted in the analysis.

In order to reliably compare data from multiple microarray chips, data were firstly  $\log_2$  transformed and then normalized by means of a widely used algorithm based on *quantile normalization* [8]. The  $\log_2$  transformation of the data is a standard step in gene expression analysis, since the usage of logarithms makes easier the analysis of expression ratios between genes [58]. The need for data normalization, instead, arises naturally when dealing with experiments involving multiple arrays in order to deal with unwanted variation that is introduced in data during the process of carrying out the experiment (i.e. due to differences in sample preparation, in the production of the arrays and in their processing).





**Fig. 7.14** Examples of gene expression profiles found during the analysis. In the first row are depicted two chaotic expression profiles, that is, time series where expression levels are not time dependent. The expression profile depicted on the right exhibits a high  $\log_2$  fold-change, this time series is usually not filtered out by standard filtering algorithms, even if its evolution is clearly not related to the experiment under consideration. In the second row, instead, some examples are given of time series whose expression profiles are time dependent. In particular, starting from the left, we have an example of a  $\nearrow$  profile, a  $\cap$  profile, and a  $\sim$  profile [48]

Normalized data have been then processed by a filtering procedure specifically developed for this work. A standard way for filtering genes consists into calculating the maximum  $\log_2$  fold-change for each gene<sup>7</sup> and then by considering only those genes which have a  $\log_2$  fold-change greater than a suitable threshold value (usually between 1 and 2). This procedure, however, filters out genes that do not change at all in time, but very often does not remove time series which evolve in time in a way that is not time dependent, especially when these time series exhibit a high  $\log_2$  fold-change (see Fig. 7.14 for some examples). In order to cope with this problem, we introduced a filtering procedure based on polynomial fitting techniques and statistical F-tests [48]. The idea behind this algorithm is based on the following assumption: if some genes are regulated by the pathway under examination, then their expression profiles must change in a time dependent way and they need to exhibit a dynamics that can be approximated by a polynomial model of degree 1 (linear  $\nearrow$  and  $\searrow$  expression profiles), 2 (parabolic  $\cup$  and  $\cap$  expression profiles) or 3 (cubic  $\sim$  and  $\simeq$  expression profiles).

The filtering procedure has been applied to more than 24,000  $\log_2$  transformed and quantile normalized time series and returned a subset of 1,175 genes, which were estimated to be HER-2 oncogene-regulated (see the second row of Fig. 7.14

<sup>7</sup> Since our time series have been  $\log_2$  transformed, the  $\log_2$  fold-change of one time series is given by the subtraction of the maximum expression value with the minimum one.

for some examples of expression profiles considered by our procedure). However, filtered genes are of the order of thousands, a number which is already too big if we want to singularly analyse each gene expression profile. For this reason, a clustering phase is needed in order to divide the genes in a set of classes which collect genes that have very similar expression profiles. In this way the MP modelling of the phenomenon will consider only an expression profile for each cluster (called *cluster profile*) calculated by averaging the time series of the gene expression profiles which are in the same group. This approach is called *guilt by association* [33] and assumes that genes with similar expression patterns are functionally related to each other.

Within literature many different cluster algorithms have been defined, most of them especially written to solve the problem of clustering gene expression time series [14]. In our application we adopted a standard hierarchical clustering technique [28] using as distance measure the point-wise difference between the time series of the derivative of gene polynomial models calculated during the filtering [48]. We considered derivatives because we focused our attention on the variation of the  $\log_2$  expression level. In fact, this amount should be directly related to the perturbing factor acting on the system during the micorarray analysis.

With this, the analysis arrived at 8 clusters. Their profiles were then used to calculate, by means of LGSS, the MP model and gene network, which suggests the regulatory interactions among the clusters. Finally, based on the identity and functional attributes of individual genes mapped to the network, the MP model pointed toward previously unknown circuits and intermediary factors in HER-2-directed signalling. A subsequent investigation of the effects on breast cancer cells due to treatment with specific inhibitors targeting the genes of the discovered new circuits demonstrated which of them are necessary for transducing HER-2 signals and support the cancer-specific growth phenotype. This demonstrates a method for uncovering which specific genes have an important role in cancer.

## 7.4 Conclusion and Ongoing Research

MP systems, based on Păun's P systems, were introduced for modelling metabolic systems by means of suitable multiset rewriting grammars. In this work, two applications of MP systems for discovering the internal regulation logic of phenomena relevant to systems biology have been presented. From the web page [1], the MetaPlab software can be downloaded, where MP grammars of Fig. 7.6 (and possible variations) can be simulated and the results presented in the paper can be confirmed.

Models of both phenomena are currently under development in order to extend the MP methodology in cases more complex (e.g. the insulin-glucose dynamics where C-peptide time series are taken into account) or other kind of gene expression analysis related to other pathological situations. However, our initial results suggest that MP systems may be considered as a suitable framework for modelling biological dynamics. In fact, the discrete time approach, which is at the basis of the MP theory, and the regression algorithm LGSS, which automatically solves the dynami-

cal inverse problem in the MP framework, make MP systems a valuable competitor of ODE systems. This is particularly true when the considered phenomenon have some regulations which act with delay or in cases where the knowledge about the phenomenon is so poor that no clear idea is available about the kind of model underlying the observed behaviour. In this perspective we intend to develop algorithmic and computational tools for making the MP modelling even more adequate and useful in biomedical applications.

**Acknowledgments** The research presented in this paper had a substantial benefit from two collaborations, which are active since 2011, with the Department of Medicine of the University of Verona, Section of Endocrinology, in particular with Prof. Riccardo Bonadonna, Dr. Maddalena Trombetta, Marco Dauriz, and Maria Linda Boselli and with the Barbara Ann Karmanos Cancer Institute and the Department of Oncology of the Wayne State University, Detroit, MI, USA.

## References

1. The MetaPlab web page. <http://mplab.sci.univr.it/>
2. A. Aczel, J. Sounderprandian, *Complete Business Statistics* (Mc Graw Hill, International Edition, 2006)
3. J. Bailey, Mathematical modeling and analysis in biochemical engineering: past accomplishments and future opportunities. *Biotechnol. Prog.* **14**, 8–20 (1998)
4. R. Bergman, D. Finegood, M. Ader, Assessment of insulin sensitivity in vivo. *Endocr. Rev.* **6**(1), 45–86 (1985)
5. R. Bergman, Y. Ider, C. Bowden, C. Cobelli, Quantitative estimation of insulin sensitivity. *Am. J. Physiol. Endocrinol. Metab.* **236**(6), 667–677 (1979)
6. G. Bocharov, F. Rihan, Numerical modelling in biosciences using delay differential equations. *J. Comput. Appl. Math.* **125**, 183–199 (2000)
7. H. Bolouri, E. Davidson, Modeling transcriptional regulatory networks. *BioEssays* **24**(12), 1118–1129 (2002)
8. B. Bolstad, R. Irizarry, M. Astrand, T. Speed, A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* **19**(2), 185–193 (2003)
9. A. Boutayeb, A. Chetouani, A critical review of mathematical models and data used in diabetology. *Biomed. Eng. Online* **5**, 43 (2006)
10. P. Brazhnik, A. de la Fuente, P. Mendes, Gene networks: how to put the function in genomics. *Trends Biotechnol.* **20**(11), 467–472 (2002)
11. H. Cao, F. Romero-Campero, S. Heeb, M. Cámara, N. Krasnogor, Evolving cell models for systems and synthetic biology. *Syst. Synth. Biol.* **4**(1), 55–84 (2010)
12. S. Choi. Introduction to Systems Biology. Humana Press (2007).
13. C. Cobelli, E. Renard, B. Kovatchev, Artificial Pancreas: Past, Present, Future. *Diabetes* **60**(11), 2672–2682 (2011)
14. I. Costa, F. de A.T. de Carvalho, M. de Souto. Comparative analysis of clustering methods for gene expression time course data. *Genet. Mol. Biol.* **27**(4), 623–631 (2004)
15. J. Cushing, in *Lecture notes in biomathematics*. Integro-differential equations and delay models in population dynamics, vol. 20. Springer-Verlag, Berlin (1977)
16. N. Draper, H. Smith, *Applied Regression Analysis*, 2nd edn. (Wiley, New York, 1981)
17. J. Fisher, T. Henzinger, Executable cell biology. *Nat. biotechnol.* **25**(11), 1239–1249 (2007)
18. A.D. Gaetano, O. Arino, Mathematical modelling of the intravenous glucose tolerance test. *J. Math. Biol.* **40**(2), 136–168 (2000)

19. J. Gerich, Redefining the clinical management of type 2 diabetes: matching therapy to pathophysiology. *Eur. J. Clin. Invest.* **32**, 46–53 (2002)
20. A. Gilman, A. Arkin, Genetic “code”: representations and dynamical models of genetic components and networks. *Annu. Rev. Genomics Hum. Genet.* **3**, 341–369 (2002)
21. P. Gilon, M. Ravier, J. Jonas, J. Henquin, Control mechanisms of the oscillations of insulin secretion in vitro and in vivo. *Diabetes* **51**(1), S144–S151 (2002)
22. A. Goldbeter, A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *PNAS* **88**(20), 9107–9111 (1991)
23. J. Hasty, D. McMillen, F. Isaacs, J. Collins, Computational studies of gene regulatory networks: in numero molecular biology. *Nat. Rev. Genet.* **2**(4), 268–279 (2001)
24. A. Heath, L. Kavraki, Computational challenges in systems biology. *Comput. sci. rev.* **3**(1), 1–17 (2009)
25. R. Hocking, The Analysis and Selection of Variables in Linear Regression. *Biometrics* **32**, 1–50 (1976)
26. S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle. COPASI-a COMplex PATHway Simulator. *Bioinformatics* **22**, 3067 (26)
27. T. Ideker, T. Galitski, L. Hood, A new approach to decoding life: Systems biology. *Annu. Rev. Genomics Hum. Genet.* **2**, 343–372 (2001)
28. S. Johnson, Hierarchical Clustering Schemes. *Psychometrika* **2**, 241–254 (1967)
29. H. Jong, Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.* **9**(1), 69–105 (2002)
30. H. Kitano, Computational systems biology. *Nature* **420**, 206–210 (2002)
31. H. Kitano, Systems biology: a brief overview. *Science* **295**(5560), 1662–1664 (2002)
32. A.D. la Fuente, P. Brazhnik, P. Mendes, Linking the genes: inferring quantitative gene networks from microarray data. *Trends Genet.* **18**(8), 395–398 (2002)
33. D. Lockhart, E. Winzler, Genomics, gene expression and DNA microarrays. *Nature* **405**, 827–836 (2000)
34. T. Maiwald, J. Timmer, Dynamical modeling and multi-experiment fitting with PottersWheel. *Bioinformatics* **24**(18), 2037–2043 (2008)
35. A. Makroglou, J. Li, Y. Kuang, Mathematical models and software tools for the glucose-insulin regulatory system and diabetes: an overview. *Appl. Numer. Math.* **56**(3), 559–573 (2006)
36. V. Manca, The metabolic algorithm for P systems: principles and applications. *Theoret. Comput. Sci.* **404**, 142–155 (2008)
37. V. Manca. Algorithmic Bioprocesses, chapter 28: Log-Gain Principles for Metabolic P Systems. *Natural Computing*. pp. 585–605. Springer-Verlag (2009)
38. V. Manca. Fundamentals of Metabolic P Systems. In [55], chapter 19 (2010).
39. V. Manca, *Infobiotics: information in biotic systems* (Springer, Berlin, 2013)
40. V. Manca, L. Bianco, Biological networks in metabolic P systems. *BioSystems* **91**(3), 489–498 (2008)
41. V. Manca, L. Bianco, F. Fontana, Evolutions and oscillations of P systems: theoretical considerations and application to biological phenomena. *LNCS* **3365**, 63–84 (2005)
42. V. Manca, L. Marchetti, Goldbeter’s mitotic oscillator entirely modeled by MP systems. *LNCS* **6501**, 273–284 (2010)
43. V. Manca, L. Marchetti, Metabolic approximation of real periodical functions. *J. Logic Algebraic Program.* **79**, 363–373 (2010)
44. V. Manca, L. Marchetti, Log-gain stoichiometric stepwise regression for MP systems. *Int. J. Found. Comput. Sci.* **22**(1), 97–106 (2011)
45. V. Manca, L. Marchetti, Solving dynamical inverse problems by means of metabolic P systems. *BioSystems* **109**, 78–86 (2012)
46. V. Manca, L. Marchetti, An algebraic formulation of inverse problems in MP dynamics. *Int. J. Comput. Math.* **90**(4), 845–856 (2013)
47. V. Manca, L. Marchetti, R. Pagliarini, MP modelling of glucose-insulin interactions in the intravenous glucose tolerance test. *Int. J. Nat. Comput. Res.* **2**(3), 13–24 (2011)

48. L. Marchetti, V. Manca. A methodology based on MP theory for gene expression analysis. CMC 2011, LNCS, **7184**, 300–313 (2012)
49. A. Mari, Mathematical modeling in glucose metabolism and insulin secretion. Curr. Opin. Clin. Nutr. Metab. care **5**(5), 495–501 (2002)
50. A. Mukhopadhyay, A. D. Gaetano, O. Arino. Modelling the intravenous glucose tolerance test: A global study for single-distributed-delay model. Discrete and Continuous Dynamical Systems - Series B (DCDS-B), **4**, 2, (2004), 407–417.
51. National Diabetes Data Group, Classification and diagnosis of diabetes mellitus and other categories of glucose intolerance. Diabetes **28**(28), 1039–1057 (1979)
52. J. Orth, I. Thiele, B. Palsson, What is flux balance analysis? Nat. Biotechnol. **28**, 245–248 (2010)
53. S. Panunzi, P. Palumbo, A. De Gaetano, A discrete single-delay model for the intra-venous glucose tolerance test. Theor. Biol. Med. Modell. **4**(35), 1–16 (2007)
54. G. Păun, *Membrane Computing* (Springer, Berlin, 2002)
55. G. Păun, G. Rozenberg, *Oxford Handbook of Membrane Computing* (Oxford University Press, Oxford, 2010)
56. K. Pearson, Notes on the history of correlation. Biometrika **13**(1), 25–45 (1920)
57. C. Priami, Algorithmic systems biology. Commun. ACM **52**, 80–88 (2009)
58. J. Quackenbush, Microarray data normalization and transformation. Nat. genet. suppl. **32**, 496–501 (2002)
59. J. Schellenberger, R. Que, R. Fleming, I. Thiele, J. Orth, A. Feist, D. Zielinski, A. Bordbar, N. Lewis, S. Rahmanian, J. K. J., D. Hyduke, B. Palsson. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. Nature protocols **6**(9), 1290–1307 (2011)
60. P. Smolen, D. Baxter, J. Byrne, Modeling transcriptional control in gene networks: Methods, recent results, and future directions. Bull. Math. Biol. **62**(2), 247–292 (2000)
61. G. Toffolo, R. Bergman, D. Finegood, C. Bowden, C. Cobelli, Quantitative estimation of beta cell sensitivity to glucose in the intact organism: a minimal model of insulin kinetics in the dog. Diabetes **29**(12), 979–990 (1980)
62. M. Trombetta, L. Boselli, A. Cretti, A. Calì, M. Vettore, B. Caruso, R. Dorizzi, A. Avogaro, M. Muggeo, E. Bonora, R. Bonadonna. Type 2 diabetes mellitus: A disease of the governance of the glucose-insulin system An experimental metabolic control analysis study. Nutrition, Metabolism & Cardiovascular Diseases. In press.
63. M. von Stosch, J. Peres, S.F. de Azevedo, R. Oliveira, Modeling biochemical networks with intrinsic time delays: a hybrid semi-parametric approach. BMC Sys. Biol. **4**, 131 (2010)
64. B. Wilhelm, J. Landry, Rna-seq-quantitative measurement of expression through massively parallel rna-sequencing. Methods **48**, 249–257 (2009)

## Chapter 8

# Modelling and Analysis of *E. coli* Respiratory Chain

Adrian Țurcanu, Laurențiu Mierlă, Florentin Ipate, Alin Stefanescu,  
Hao Bai, Mike Holcombe and Simon Coakley

**Abstract** In this chapter we present some results obtained in the study of the bacterium *E. coli* related to its behavior at different level of oxygen in the environment. The biological model is expressed in terms of different molecules and their reactions. First, an agent-based model of *E. coli* is implemented in the FLAME framework for multi-agents and some simulation results are given. Each agent is represented by an X-machine and the model corresponds to communicating X-machines. Then this model is transformed into a kernel P system. This kernel P system is implemented in the Rodin platform and in Spin and some properties are verified using the associated model checkers. Formulated using the LTL formalism, the verified properties refer to the variation of the number of different molecules as a result of the occurring reactions. Our main contribution is a simplified model of *E. coli* that preserves the main properties of the initial model, and can be formally verified using a model checker.

---

A. Țurcanu (✉) · L. Mierlă · F. Ipate · A. Stefanescu  
Department of Computer Science, University of Pitești, Pitești, Romania  
e-mail: adrianturcanu85@yahoo.com

L. Mierlă  
e-mail: laurentiu.mierla@gmail.com

A. Stefanescu  
e-mail: alin@stefanescu.eu

F. Ipate  
Department of Computer Science, University of Bucharest, Bucharest, Romania  
e-mail: florentin.ipate@ifsoft.ro

H. Bai · M. Holcombe · S. Coakley  
Department of Computer Science, University of Sheffield, Sheffield, UK  
e-mail: hao.bai@sheffield.ac.uk

M. Holcombe  
e-mail: m.holcombe@epigenesys.co.uk

S. Coakley  
e-mail: s.coakley@sheffield.ac.uk

## 8.1 Introduction

Membrane computing, a research field introduced in 1998 by Gheorghe Păun, studies computing devices inspired by the functioning and structure of the living cell, called P systems [33]. Since the appearance of membrane computing, many variants of P systems have been defined and investigated, particularly in terms of computational power and their capability to solve computationally hard problems [34]. Introduced in [18], kernel P systems (kP systems), and its reduced variant [20] (skP systems) represent an unifying framework for P systems which integrates many features of existing P systems into an elegant and yet powerful modelling formalism.

In the last years, significant developments have also been made in using P systems to model, simulate and formally verify various systems [13]. As a consequence, the idea of automating the evolution of a P system was one of the main concerns of the membrane computing community and many tools have been developed, as surveyed in [15]. However, each of these tools came with its own specification language, and was dedicated to a certain class of P systems. In this context, the Research Group of Natural Computing, from the University of Seville, developed P-Lingua [16], a programming language for P systems that became the standard for their representation. Another useful tool developed by the same research group is MeCoSim [32], a membrane computing simulator that can be easily adapted to each family of P systems. Using a P-Lingua definition file of a P system, MeCoSim can be used for simulations and property extraction.

Formal verification of biological systems has been studied using e.g., rewriting logic and the Maude tool [9] or PRISM and the associated probabilistic temporal logic [21] for stochastic systems [11]. More recently, various properties of transition P systems, P systems with active membranes and kernel P systems have been verified using different tools like NUSMV [19], SPIN [27] and RODIN [28, 36].

Based on the X-machine formalism [22], FLAME (Flexible Large-scale Agent Modelling Environment) [1, 35] is a generic agent-based modelling system, which can be used to develop and simulate applications in many areas. In particular, the FLAME framework has been proven very successful for modelling and simulating different biological, economic and social systems. In FLAME, each agent has a memory that holds variables and evolves according to a transition diagram in which the transitions are labelled by processing functions. These functions can read and write to variables in the agent's memory or can read incoming messages and write outgoing messages. The agents communicate via messages. One of the great strengths of FLAME is its modularization and platform independence, which allows it to be run on parallel supercomputers.

One of the success stories of FLAME is the SUMO (Systems Understanding of Microbial Oxygen Responses) research project [5], funded by the European research initiative SysMO (System Biology of Microorganisms), centered around the *Escherichia coli* (*E. coli*) bacterium, one of the most studied organism in biology [23]. In particular, the project investigates the behavior of this bacterium related to its reaction to the level of oxygen in the environment. Using the mathematical models



and simulation results provided by FLAME, SUMO deepened the existing knowledge about the metabolic adaptation that occurs in response to changes in oxygen availability [5].

A number of recent investigations [26, 20] illustrate the expressive power of kP systems on several case studies, representing known NP-complete problems. This chapter makes a further step in this direction, by using the kP system as a modelling tool for biological systems. To this end, we show how the X-machine based models of the *E. coli* developed in FLAME can be naturally transformed into kP system models. Furthermore, these kP systems also provide the basis for the implementation in two modelling languages (Event-B and Promela) and the associated model checkers, ProB [3, 30] and Spin [4, 24], are used to simulate and verify some of their properties. In related chapters, the probabilistic model checking tool PRISM was used for formal verification in systems biology for quantitative properties [29] or stochastic trend formulae [8]. Introduced in [12], the Infobiotics Workbench is also a powerful computational framework incorporating model specification, simulation, parameter optimisation and model checking for various systems biology problems. Finally, BMA tool [10] makes formal methods accessible to biologists by means of an intuitive visual interface.

The chapter is structured as follows. In the next section we overview the theoretical models of kP systems and X-machines. We continue by describing *E. coli* and its respiratory chain in Sect. 8.3 and its FLAME simulation in Sect. 8.4. Section 8.5 shows how to use the kP systems to capture interesting and relevant aspects of *E. coli* behaviour. The verification of different properties is implemented using two different model-checkers in Sect. 8.6. Conclusions are drawn in the last section.

## 8.2 Background

In this section we provide the formal definitions of a variant of P systems and of X-machines.

### 8.2.1 kP Systems

P systems are distributed and parallel computing devices processing multisets of objects encapsulated into regions delimited by membranes, using various types of rules (evolution, communication, division and others). Kernel P systems provide an unifying framework for many features available in various P system variants [18]. In this chapter we will use a simplified version of kernel P systems called simple kernel P systems [20].

**Definition 1** A *simple kernel P system* (skP system, for short) of degree  $n \geq 1$  is a tuple



$$sk\Pi = (A, L, IO, C_1, \dots, C_n, \mu, i_0)$$

where:

- $A$  is an alphabet containing objects,
- $L$  is a finite set of labels,
- $IO$  is an alphabet,  $IO \subset A$ , associated with the environment,
- $C_1, \dots, C_n$  are the initial compartments of the system; each of them is identified by a label of  $L$ , has initially a multiset over  $A$  and a finite set of rules,
- $\mu = (V, E)$  is an undirected graph, where  $V \subseteq L$  are vertices and  $E$  the edges, and
- $i_0 \in L \cup \{0\}$  denotes the *output region*, i.e., the compartment receiving the result of a computation.

A skP system  $(A, L, IO, C_1, \dots, C_n, \mu, i_0)$  can be viewed as a set of  $n$  compartments  $C_1, \dots, C_n$ , interconnected by edges from  $E$  of an undirected graph  $\mu$ . Every compartment has an associated set of rules that can be of type division, rewriting or communication. Rules may have guards (necessary conditions) and are applied in maximally parallel mode. The only restrictions are that at most one division rule can be applied per step and, when a cell is divided, the division rule is the only one which is applied for that cell in that step. We describe the guards and the rules syntax below.

The guards are constructed using multisets over  $A$ , relational and Boolean operators. Before defining it, we introduce some notations. For a multiset  $w$  over  $A$  and an element  $a \in A$ , we denote by  $|w|_a$  the number of  $a$ 's occurring in  $w$ . Let  $Rel = \{<, \leq, =, \neq, \geq, >\}$  be the set of relational operators,  $\gamma \in Rel$  a relational operator,  $a^n$  a multiset and  $r\{g\}$  a rule with guard  $g$ .

**Definition 2** If  $g$  is the abstract relational expression  $\gamma a^n$  and the current multiset is  $w$ , then the guard is true for the multiset  $w$  if  $|w|_a \gamma n$  is true.

Abstract relational expressions can be connected by Boolean operators ( $\neg$ ,  $\wedge$  and  $\vee$ ) generating *abstract Boolean expressions*.

**Definition 3** If  $g$  is the abstract Boolean expression and the current multiset is  $w$ , then the guard denotes the Boolean expression for  $w$ , obtained by replacing abstract relational expressions with relational expressions for  $w$ . The guard  $g$  is true for the multiset  $w$  when the Boolean expression for  $w$  is true.

**Definition 4** A guard is defined recursively as:

- (i) one of the Boolean constants *true* or *false*, or
- (ii) an abstract relational expression, or
- (iii) an abstract boolean expression.

*Example.* The guard  $g = \geq a^3 \wedge \geq b^4 \vee \neg > c$  is true for  $w$  if  $w$  contains at least 3  $a$ 's and 4  $b$ 's or no more than one  $c$ .

The rules can have one the following syntax:

- (a) rewriting and communication rule:  $x \rightarrow y \{g\}$ ,  
 where  $x \in A^+$ ,  $y \in A^*$ . The right hand side  $y$ , has the form  $y = (a_1, t_1) \dots (a_h, t_h)$ , where  $a_j \in A$  and  $t_j \in L$ ,  $1 \leq j \leq h$ , is an object and a target, i.e., the label of a compartment, respectively, and  $(a_i, t_i) \neq (a_j, t_j)$ , for each  $1 \leq i, j \leq h$ ,  $i \neq j$ . The target  $t_j$ , must be either the label of the current compartment, say  $l_i$ , (most often ignored) or that of an existing neighbour of it  $((l_i, t_j) \in E)$  or an unspecified one,  $*$ ; otherwise, the rule is not applicable. If a target  $t_j$  refers to a label that appears more than once, then one of the involved compartments will be non-deterministically chosen. If  $t_j$  is  $*$  then the object  $a_j$  is sent to a neighbouring compartment arbitrarily chosen.  
*Example.* If the rule is  $r : a \rightarrow a(b, 2)(c, 3) \{g\}$ , then it is applicable iff the guard  $g$  is true, and, as a result of its application, one object  $a$  stays in the current compartment (we do not use target for it), one object  $b$  is sent to the compartment labelled 2 and one object  $c$  is sent to the compartment labelled 3.
- (b) membrane division rule:  $[x]_{l_i} \rightarrow [y_1]_{l_{i_1}} \dots [y_h]_{l_{i_h}} \{g\}$ , where  $x \in A^+$  and  $y_j = (a_{j,1}, t_{j,1}) \dots (a_{j,h_j}, t_{j,h_j})$ , where  $a_{j,k} \in A$ ,  $t_{j,k} \in L$ , and  $1 \leq k \leq h_j$ . In this case, the compartment  $l_i$  will be replaced by  $h$  compartments, with the labels  $l_{i_1}, \dots, l_{i_h}$ ; furthermore, for  $1 \leq j \leq h$ , the  $i_j$ -th compartment will contain the same objects as  $l_i$  with the exception of  $x$ , which will be replaced by  $y_j$ . Moreover, all the links of  $l_i$  are inherited by each of the newly created compartments.  
*Example.* If the rule is  $r : []_2 \rightarrow []_{21}[]_{22}[]_{23} \{g\}$ , then it is applicable iff the guard  $g$  is true, and, as a result, the compartment with label 2 is replaced with 3 compartments with the same content as compartment 2.

In our models only rewriting and communication rules will be used. Furthermore, all such rules are non-cooperative, i.e., only one object appears on the left side of each rule.

Since we are dealing with the model of a biological system, probabilities are added to the rules. The idea of adding probabilities to P systems was based on the intention to keep membrane system theory as close as possible to the biological reality [31].

We consider that probabilistic rules are complementary, i.e., these can be grouped into sets (pairs in our case) with the same left side and the sum of the probabilities associated with the rules of each pair being 100 %. Thus, if  $r_1 : x \xrightarrow{p\%} y$  and  $r_2 : x \xrightarrow{(100-p)\%} z$  are two probabilistic rules, then rule  $r_1$  is applied with a probability of  $p\%$  and rule  $r_2$  with a probability of  $(100 - p)\%$ . After being associated with the objects according to their probabilities, rules are applied in a maximal parallel manner in each compartment. (The difference between the original, non-deterministic, kP system and the probabilistic one is that, at each moment within a maximally parallel computation step, whenever more than one rule can be selected, the applied rule is selected according to its probability rather than non-deterministically.)

### 8.2.2 X-Machines

Introduced in 1974 by Samuel Eilenberg [17], X-machines were proposed as a basis for a possible specification language by Mike Holcombe in 1988 [22]. They provide the modelling foundation for the FLAME framework. X-machines are computational models that can describe a system as a finite set of states, each with an internal store called memory, and a number of transitions between the states. A transition is triggered by an input value, produces an output value and may alter the memory. An X-machine may be modelled by a finite automaton in which the arcs are labelled by function names (the *processing functions*).

**Definition 5** An *X-Machine* is a tuple

$$XM = (\Sigma, \Gamma, Q, M, \Phi, F, I, T, m_0),$$

where:

- $\Sigma$  and  $\Gamma$  are finite sets called *input alphabet* and respectively *output alphabet*,
- $Q$  is the finite set of *states*,
- $M$  is a (possibly) infinite set called *memory*,
- $\Phi$  is the *type* of  $XM$ , a non-empty finite set of *function symbols*. A *basic processing function*  $\varphi : M \times \Sigma \longrightarrow \Gamma \times M$  is associated with each function symbol  $\varphi$ .
- $F$  is the (partial) *next state function*,  $F : Q \times \Phi \longrightarrow 2^Q$ ,  
As for finite automata,  $F$  is usually described by a *state-transition diagram*.
- $I$  and  $T$  are the sets of initial and terminal states respectively,  $I \subseteq Q$ ,  $T \subseteq Q$ , and
- $m_0 \in M$  is the initial memory value.

**Definition 6** A *communicating X-machine system* [25] with  $n$  components is a tuple  $S_n = ((XM_i)_{1 \leq i \leq n}, R)$ , where:

- $XM_i = (\Sigma_i, \Gamma_i, Q_i, M_i, \Phi_i, F_i, I_i, T_i, (m_0)_i)$  is an X-machine labelled by  $i$ , for  $1 \leq i \leq n$ , and
- $R$  is a relation defining the communication among the components,  
 $R \subseteq \{XM_1, XM_2, \dots, XM_n\} \times \{XM_1, XM_2, \dots, XM_n\}$ . A tuple  $(XM_i, XM_j) \in R$ , denotes that the X-machine  $XM_i$  can output a message to a corresponding input stream of X-machine  $XM_j$ , for any  $i, j \in \{1, \dots, n\}$  with  $i \neq j$ .

The exchange of messages among the components of a communicating X-machine is achieved by redirecting one component's function output to be received as input by a function of another machine.

## 8.3 General Description of *E. coli*

*E. coli* is one of the most studied bacterium and the research related to it provided many fundamental paradigms in biology. It can be easily handled by biologists and

its genetic information and metabolic processes are well understood. Interestingly, unlike many organisms, *E. coli* can thrive in environments either with abundant oxygen or no oxygen.

A system-level study of the mechanism of *E. coli* responding to oxygen is a key to understand the respiratory pathways of this bacterium. Based on FLAME, an agent-based model was introduced to better understand the respiratory chain and to simulate the activities of relative components, such as oxidases *Cyo* and *Cyd*, and their regulators, *Fnr* and *ArcA*. In this model, the expression of *Cyd* and *Cyo* in *E. coli* are repressed or activated by *Fnr* and *ArcA*. An integration of COPASI (a software application for simulation and analysis of biochemical networks and their dynamics) makes it possible to calculate the dynamic variation of *Cyo/Cyd* numbers using mathematical methods. Compared with the traditional kinetic models, which consider the system as a macroscopic quantity, the agent-based model represents every single molecule and enables the activities of agents in an actual spatial region. This advantage of agent-based model could crucially complement the kinetic model, for the cases when the latter fails, such as low molecule number or unevenly distributed molecules.

In this agent-based model, each individual molecule of interest is defined as an agent with its own parameters, such as position, status etc. According to the biochemical reaction conditions, these agents could exist within the cellular environment and interact with each other. This cellular space can be defined as a 2-dimensional or 3-dimensional space, in which the molecules may be close to the membrane or evenly distributed in cytoplasm. All these molecules are capable of moving through this space and interacting with others according to their interaction radius. The current agent model consists of the following types of agents: oxygen (*O2* molecules), *E. coli* cell, *Fnr* molecules, *ArcB* molecules, *ArcA* molecules, *ArcBA* molecules, *Cyo* and *Cyd* molecules. This is a preliminary model, but we are currently working with a team of biologists on an improved version much closer to the real biological system.

The FLAME framework allows to define agents in a precise way and stores their initial specifications in an XML file. On reading this file, the pre-defined agents are constructed in a virtual space for further activities. Each agent communicates with the others via message boards. These messages contain information on the whereabouts and state of the molecule. Together with a random moving algorithm and pre-defined interaction rules, this information drives the whole model and leads to a final output. In the current model, the *Fnr* molecules are divided into three groups based on their status, including *Fnr* dimer, *Fnr* monomer and *Fnr* dimer bound to the binding sites on DNA. The mechanism of interaction between *Fnr* and oxygen molecule is defined as follows:

- When an oxygen molecule is within a pre-defined reaction distance to an *Fnr* dimer, the *Fnr* dimer is decomposed into two *Fnr* monomers. If this dimer is bound to a binding site, the binding site will become unoccupied.
- When two *Fnr* monomers are within reaction distance, they can be combined into an *Fnr* dimer.

- When the distance between a *Fnr* dimer and an unoccupied binding site is less than their reaction distance, the dimer will bind to the binding site.

The ArcBA molecules are present as *ArcA* octamers, *ArcA* tetramers, *ArcA* dimers, *ArcB* and *ArcA* octamers bound to the binding sites on DNAs. The *ArcA* octamers bound to binding sites have a probability of coming off. The *ArcB* molecules can be phosphorylated (*ArcB* P) or dephosphorylated (*ArcB*), depending on how much it was exposed to oxygen. The *ArcA* dimers can also be in these two forms. All the *ArcA* octamers and tetramers contain phosphorus. The mechanism of interaction between *ArcBA* molecules and oxygen is defined as:

- When an oxygen molecule is within pre-defined reaction distance to an *ArcB*, the *ArcB* will be de-phosphorylated and then become able to capture phosphorus from active *ArcA* dimers, *ArcA* tetramers and *ArcA* octamers. When an *ArcB* has no oxygen molecule to interact with, the capability of this *ArcB* of capturing phosphorus will reduce proportionally by revising its reaction radius.
- When an *ArcA* octamer is moving into the reaction distance to *ArcB*, the *ArcA* octamer will be de-phosphorylated and decomposed into one *ArcA* tetramer, one active *ArcA* dimer and one inactive *ArcA* dimer.
- When an *ArcA* tetramer is moving into the reaction distance to *ArcB*, the *ArcA* tetramer will be de-phosphorylated and decomposed into one active *ArcA* dimer and one inactive *ArcA* dimer.
- When an active *ArcA* dimer is moving into the reaction distance to *ArcB*, the *ArcA* dimer will be de-phosphorylated.
- When an inactive *ArcA* dimer is moving into the reaction distance to *ArcB* P, the *ArcA* dimer will be phosphorylated.
- When two active *ArcA* dimers are within their reaction distance, they can be combined into an *ArcA* tetramer.
- When two *ArcA* tetramers are within their reaction distance, they can be combined into an *ArcA* octamer.
- When one *ArcA* octamer is within its reaction distance to an available binding site, the *ArcA* octamer will bind to the binding site.

## 8.4 FLAME Simulations of *E. coli* Respiratory Chain

As we stated before, the *E. coli* FLAME model is based on agents corresponding to the molecules. At every step in the simulation, each molecule must change its location (move) according to predefined rules. The size of the molecules is assumed to be sufficiently small that collision between them can be neglected in the movement process.

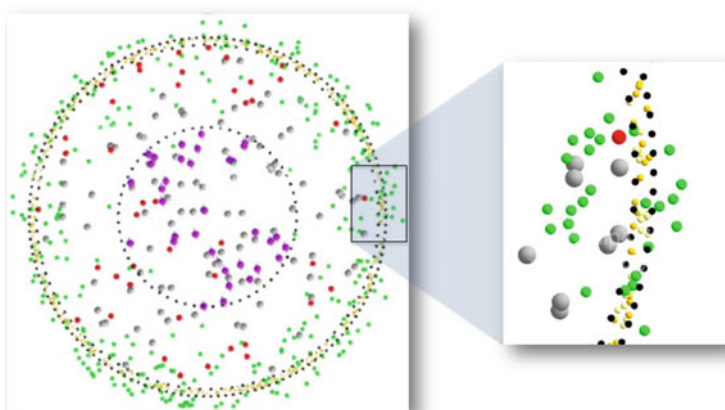
Each agent is represented by an X-machine whose memory contains an ID, the type of the agent, the physical location and its state. The behavior of each molecule is modelled in the rules of the corresponding agent: namely, with what molecules it can

interact, and what distance is necessary for this interaction to occur. Agents communicate by sending and receiving messages containing values from their memory or announcing their availability to interact. Each agent contains a function that calculates the distance to all the other agents by using its coordinates and the coordinates received from other agents through messages. Besides the criteria on proximity (i.e., the calculated distance is less than the pre-defined interaction radius), two molecules interact if they are in a state that allows interactions. Each X-machine also contains functions that compute the total number of molecules of that type, move the molecules, send availability to other agents or destroy the agent if it is consumed in a reaction. These functions also change the state of the X-machine accordingly. Thus, the model corresponds to a communicating X-machine.

Since we are interested only in some of the results of the FLAME simulations, in this section we consider a simplified version of *E. coli* cell, with only eight molecules types: dimers, regenerated dimers, monomers, oxygen, dimers bound to binding sites, *Cyo* proteins, *Cyd* proteins, binding sites, but without the *ArcBA* molecules. Note also that FLAME provides visualisation and animation capabilities, as seen in Fig. 8.1, but we do not insist here on these aspects.

Oxygen molecules enter the cell through the membrane where a large amount of them are consumed by the oxidases (*Cyo* or *Cyd* proteins). Those which luckily get into the cell (about 1 % of them) react with the *Fnr* dimers and generate *Fnr* monomers. Two *Fnr* monomers getting close enough could react and regenerate a *Fnr* dimer. The *Fnr* dimer which was already bound to binding sites can also be deactivated by oxygen molecules and leave the binding site. Also, *Fnr* dimers can bind to available binding sites when they get close enough.

Initially, the cell is considered to be in an anaerobic respiration state (no oxygen in the cell), and the values of the molecules are: 150 dimers, 1 monomer, 35 dimers



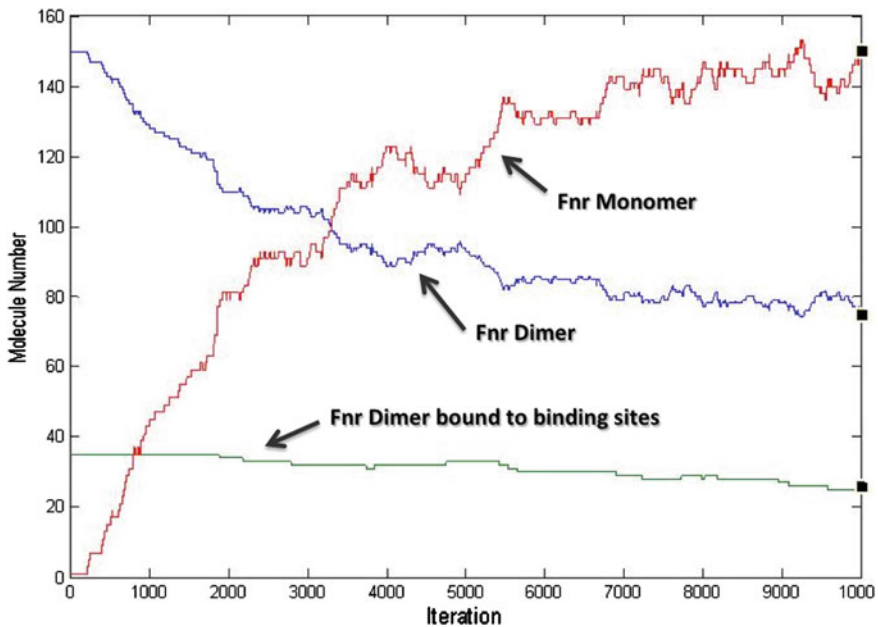
**Fig. 8.1** Visual simulation of *E. coli* in FLAME: the cell (left side) and a zoomed-in snapshot of its membrane (right side)

bound to binding sites, 200 *Cyo* proteins, 200 *Cyd* proteins, 35 binding sites. In these conditions, FLAME can be used to simulate the reaction in the cell at different levels of oxygen, e.g., 100, 200, or 300 molecules. Figure 8.2 provides the simulation results for 100 molecules of oxygen over 10,000 iterations.

Besides the relation between the numbers of oxygen, *Fnr* dimer and *Fnr* monomer molecules, more interesting results are obtained for the model that also contains the *ArcBA* system. These are related to:

- the number of *ArcA* octamer, *ArcA* tetramer, *ArcA* dimer and *ArcB* (phosphorylated and non-phosphorylated);
- the variation of the numbers of *Fnr* dimer and *ArcA* octamer which are bound to the binding sites, as these would determine the gene regulation and protein production.

So, as this model is currently used for simulation of biological processes, the number of the molecules (in biology, the concentration of certain gene regulator) is what the experiments are focusing on. More details on using FLAME for biological models, including the *E. coli* respiratory chain, can be found in [23].



**Fig. 8.2** Trend of molecule numbers with 100 oxygen molecules over 10,000 iterations

## 8.5 A Kernel P System Corresponding to *E. coli*

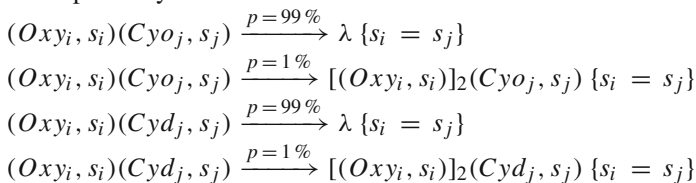
In this section, we outline the way in which the previous X-machine based model is transformed into a kernel P system of degree 2 ( $k\Pi$ ). The latter model is then used in the next section to formally verify some properties and simulate its behavior using the model checkers ProB and Spin. Due to the limitations of the corresponding modelling languages (e.g., the well-known state explosion problem) we considered a simplified model described below.

First of all, we split the *E. coli* cell into sectors (of circles) and we associate to each molecule a sector, depending on its coordinates. Thus, the objects of our P systems are pairs  $(l, s)$ , where  $l$  is the label of the object and  $s$  is the corresponding sector. The idea of using topological spaces as control mechanisms for rule applications, in addition to the membranes themselves (thus offering a higher level of granularity), was introduced in [14]. As a consequence, the alphabet of  $k\Pi$  is:  $V = \{(Oxy_i, s_i) \mid i = 1..noOxy\} \cup \{(Cyd_i, s_i) \mid i = 1..noCyd\} \cup \{(Cyo_i, s_i) \mid i = 1..noCyo\} \cup \{(Dim_i, s_i) \mid i = 1..noDim\} \cup \{(BDim_i, s_i) \mid i = 1..noBDim\} \cup \{(BSite_i, s_i) \mid i = 1..noBSites\} \cup \{(Mon_i, s_i) \mid i = 1..noMon\}$ , where *Oxy* corresponds to the Oxygen molecules, *Dim* corresponds to the Dimers, *BDim* corresponds to the dimers bound to binding sites, *BSite* corresponds to the binding sites, *Mon* corresponds to the monomers and *noX* is the number of molecules of type *X*.

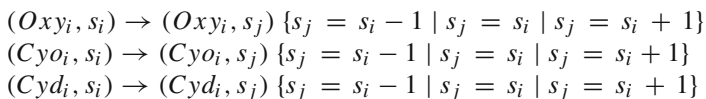
We assume that two molecules react only if they are in the same sector.

Then, the rules associated with the first compartment are:

- rewriting and communication rules corresponding to oxidations and oxygen transfer respectively:



- movement rules:



Probabilities associated with the above rules are required because, as we specified in the previous section, just about 1 % of the oxygen molecules get into the cell. Thus, an oxygen molecule  $Oxy_i$  situated in a sector  $s_i$  in the first compartment is transferred in the second compartment with a probability of 1 % or react with an oxidant situated in the same sector with a probability of 99 %. These probabilities do not appear explicitly in the FLAME model because the oxidations occur more frequently due to the molecules location (the oxidases *Cyo* and *Cyd* act like a barrier for the oxygen).



The second compartment has the following rewriting rules:

- reactions between a dimer and an oxygen resulting two monomers  
 $(Oxy_i, s_i)(Dim_j, s_j) \rightarrow (Mon_k, s_i)(Mon_l, s_i) \{s_i = s_j\}$
- reactions between a bounded dimer and an oxygen resulting a dimer and a binding site  
 $(Oxy_i, s_i)(BDim_j, s_j) \rightarrow (Dim_k, s_i)(BSite_l, s_i) \{s_i = s_j\}$
- reactions between two monomers resulting a dimer  
 $(Mon_i, s_i)(Mon_j, s_j) \rightarrow (Dim_k, s_i) \{s_i = s_j\}$
- reactions between a dimer and a binding site resulting a bounded dimer  
 $(Dim_i, s_i)(BSite_j, s_j) \rightarrow (BDim_k, s_i) \{s_i = s_j\}$
- movement rules:  
 $(Oxy_i, s_i) \rightarrow (Oxy_i, s_j) \{s_j = s_i - 1 \mid s_j = s_i \mid s_j = s_i + 1\}$   
 $(Dim_i, s_i) \rightarrow (Dim_i, s_j) \{s_j = s_i - 1 \mid s_j = s_i \mid s_j = s_i + 1\}$   
 $(Mon_i, s_i) \rightarrow (Mon_i, s_j) \{s_j = s_i - 1 \mid s_j = s_i \mid s_j = s_i + 1\}$   
 $(BDim_i, s_i) \rightarrow (BDim_i, s_j) \{s_j = s_i - 1 \mid s_j = s_i \mid s_j = s_i + 1\}$   
 $(BSite_i, s_i) \rightarrow (BSite_i, s_j) \{s_j = s_i - 1 \mid s_j = s_i \mid s_j = s_i + 1\}$

For all the above rules, guards are used to impose the constraints that two molecules react iff they are in the same sector ( $s_i = s_j$ ) and, when moving, each molecule can remain in its sector ( $s_j = s_i$ ) or can move into one of the neighboring sectors ( $s_j = s_i - 1$  or  $s_j = s_i + 1$ ).

Using these ideas and the particularities of each language, we constructed Event-B and a Promela models of  $k\Pi$ , and used them for formal verification. The obtained results are given in the next subsections.

## 8.6 Modelling, Simulation and Verification

### 8.6.1 Implementation in Event-B for ProB

Introduced by J.-R. Abrial [6], Event-B is a formal modelling language used for developing mathematical models of complex systems with a discrete behavior. It is supported by a platform called Rodin that integrates theorem-proving, model checking (ProB), and animation facilities.

The Event-B model of the kernel P system representation of *E. coli* described in the previous section is based on functions, operations with sets and non-deterministic assignments. Thus, for every set of molecules, we consider a partial function between a set of labels and the set of sectors. In the initialization event, every molecule initially in the *E. coli* cell is associated with its corresponding sector. The model contains an event for every reaction (rule), with the guards asking to the reactants to be in the same compartment, and the actions modifying the corresponding functions accordingly.

As an example, the event corresponding to the rule

$(Oxy_i, s_i)(Dim_j, s_j) \rightarrow (Mon_k, s_i)(Mon_l, s_i) \{s_i = s_j\}$  is:

```

Event DimerOxygen
any  $x, y, z_1, z_2$ 
where
  guard1:  $x \in \text{dom}(Oxy)$ 
  guard2:  $y \in \text{dom}(Dim)$ 
  guard3:  $Oxy(x) = Dim(y)$ 
  guard4:  $z_1 \in \mathbb{N} \setminus \text{dom}(Mon)$ 
  guard5:  $z_2 \in \mathbb{N} \setminus \text{dom}(Mon)$ 
  guard6:  $z_1 \neq z_2$ 
then
  action1:  $Mon := Mon \cup \{z_1 \rightarrow Oxy(x), z_2 \rightarrow Oxy(x)\}$ 
  action2:  $Oxy := Oxy \setminus \{x \rightarrow Oxy(x)\}$ 
  action3:  $Dim := Dim \setminus \{y \rightarrow Dim(y)\}$ 

```

So, if the oxygen molecule  $x$  and the dimer  $y$  are in the same compartment (guard3), then two new monomers ( $z_1$  and  $z_2$ ) are produced and added to the monomers set (action1). Then, the reactants are consumed (action2 and action3). The events for the other reactions are quite similar.

Probabilities associated with some rules are implemented using a random number generator, e.g., a random number positive integer  $N$  is generated and the rule with the probability of 1 % is applied if  $N$  is divisible by 100 and the complementary rule, otherwise.

A different type of event is the one in which the molecules are moving. We remind that any molecule can non-deterministically remain in its sector or can move into one of the neighbouring sectors.

One of main problems in model checking is the state space explosion. In order to mitigate this, we considered in our model a variable called *state* with three possible values: *Reacting*, *Moving* and *Crash*. The system is usually in one of the states *Reacting* or *Moving*. The state *Crash* is considered in order to keep the number of configurations under control. When the system reaches a number of steps (*Max*), then its state becomes *Crash* and the verification stops. We considered in our experiments different values for *Max*, depending on the complexity of the property.

The results of verifying different properties based on this Event-B model are provided in Sect. 8.6.4.

A very important Event-B concept is refinement, which allows a model to be developed gradually [7]. A refined model contains more details about the system than the initial model and it is obtained by refining machines or extending contexts. Using this technique we can add to the previous Event-B model of *E. coli* the *ArcBA* system, i.e., variables corresponding to the *ArcBA* molecules and events corresponding to the reactions between them. We describe in the following how the refined model is obtained.

As in the initial model, we consider a partial function between a set of labels and the set of sectors for every set of *ArcBA* molecules. The initialization event has to be refined, associating to every new molecule the corresponding sector. The refined model contains seven new events, corresponding to the seven reactions that involve *ArcBA* molecules described in Sect. 8.3. Each of these is similar to those in the initial model. The event dedicated to molecules movement is also refined, the *ArcBA*

molecules being also able to remain in the same sector or to move in a neighboring one. Even if the limitations of ProB, due again to state explosion, do not allow us to verify significant properties for the refined model, refinement proves to be an useful technique in building Event-B models of complex systems.

### 8.6.2 Implementation in Promela for Spin

This subsection describes the Promela implementation of the kP system model for *E. coli*. A mature model checking tool, Spin has been also successfully used in the context of membrane computing for verifying various types of P systems [27], including kernel P systems [20].

Although computationally equivalent with the investigated kP system model, the Promela implementation takes advantage of the power and flexibility of its specific modelling language constructs. For instance, a molecule is defined by using the type definition where the specific features are the *sector* where the molecule finds itself and a flag denoting whether it is still *active* or *consumed*.

```
typedef Molecule {
  int sector;
  bool isConsumed = true; }
```

Using this construct, each molecule type is assigned a global array representing the set of molecules of the same type.

The reactions between specific molecules are modelled using Promela macro-definitions. For instance, the kP system rule

$$(Mon_i, s_i)(Mon_j, s_j) \rightarrow (Dim_k, s_i)\{s_i = s_j\}$$

is computationally equivalent with the following Promela code:

```
inline MonomerMonomer() {
  skip;
  d_step {
    int x, y, foundX, foundY;
    bool found;
    found = false;

    for(x: 0..MaxMonCount - 1) {
      for(y: 0..MaxMonCount - 1) {
        if
          :: !found && Mon[x].isConsumed == false && Mon[y].
             isConsumed == false && Mon[x].
                                     sector == Mon[y].
                                     sector && x
                                     != y ->
              found = true; foundX = x; foundY = y;
          :: else -> skip;
        fi; } }

    if
      :: found ->
        createNewDim(Mon[foundX].sector);
        Mon[foundX].isConsumed = true;
    fi; }
```

```

        Mon[foundY].isConsumed = true;
    :: else -> skip;
fi;
skip; } }

```

Another well-suited Promela feature for *E. coli* modelling is the nondeterministic control statement. Throughout the model, nondeterministic guarded commands are used for simulating random molecule assignment to regions, molecule movement and reaction triggering. For instance, the change in position of *Fnr* dimer molecules are modelled using the following macro definition:

```

inline moveDim() {
    int index;
    for(index: 0..MaxDimCount - 1) {
        if
        :: Dim[index].isConsumed == false ->
            if
            :: true && Dim[index].sector > 0 -> Dim[index].sector--;
            :: true -> skip;
            :: true && Dim[index].sector < MaxSectorCount -> Dim[index]
                .sector\,+\,++;
            fi;
        :: else -> skip;
        fi; }
    index = 0; }

```

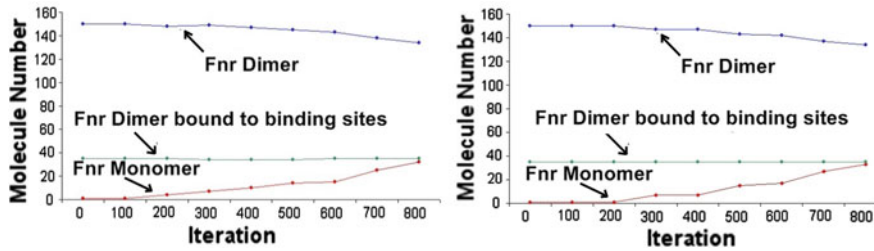
All the above macros are wrapped up in a scheduler process which is responsible for evolving the system from an initial configuration to a final one, with respect to a predefined number of steps. At each step, random molecule movement and reaction triggering is issued in order to simulate the nondeterministic behavior of *E. coli* components.

### 8.6.3 Simulation Results

In this subsection we provide some simulation results obtained for the Event-B and Promela models of *E. coli*, by comparison with the FLAME simulation results.

The number of molecules of type *X* is denoted as *noX*. As for the FLAME model, the initial values of the molecules are: *noDim* = 150, *noMon* = 1, *noBDim* = *noBSite* = 35, *noCyd* = *noCyo* = 200. In these conditions, ProB can be used to simulate the events corresponding to the reactions in the cell at different levels of oxygen. Figure 8.3 provides an average of simulation results for 100 molecules of oxygen over 800 iterations. Although the current limitations of ProB do not allow us to simulate more steps in the evolution of the Event-B model, Fig. 8.3 shows a similar variation of the molecule numbers for both models. The small differences between the two graphs are due to the approximations made in the kernel P system model.

Besides the Event-B simulations, a series of simulations have been conducted using the Promela model and Spin model checking tool, using the same initial configuration of molecule numbers and different levels of oxygen. The trend of molecule numbers was similar as for the simulations conducted using the Event-B and FLAME



**Fig. 8.3** Trend of molecule numbers with 100 oxygen molecules over 800 iterations: in Event-B (*left*) and in FLAME (*right*)

models. However, compared to ProB, Spin was able to run more iterations and performed better in terms of execution time and memory requirements.

### 8.6.4 Verification Results

We present now the different properties that we verified, mainly checking how the number of different molecules evolves during the cell reactions.

We start again with the Event-B model and ProB model checker. In the Event-B model, we introduced variables counting the number of molecules of each type, denoting again with  $noX$  the number of molecules of type  $X$ . Initially, we considered  $noOxy = noCyd = noCyo = 100$ ,  $noMon = 1$ ,  $noDim = 75$ ,  $noBdim = noBSites = 18$ .

Properties verified with ProB can be formulated using the LTL (linear temporal logic) formalism. We give some of these properties and the result given by the model checker in the following:

- $G\{noMon < 3 \text{ or } state = Crash\}$ ; the model checker returns a counterexample so, in some situations, a dimer is divided in two monomers before reaching the state Crash.
- $G\{noMon < 7 \text{ or } state = Crash\}$ ; no counterexample found, so the insertion rate of oxygen in the second compartment is very low.
- $G\{noBdim > 17\}$ ; the model checker returns a counterexample so the bounded dimers are sometimes involved in reactions.

Unfortunately, the current limitations of ProB do not allow us to verify more complicated properties or to increase the number of molecules of each type. Our future work will concentrate on improving these aspects. All the above properties were also verified with Spin, obtaining the same results. More than that, as we see below, the results obtained with the Spin model checker compensate for some cases not tackled by ProB.

In Spin, we have also used LTL to specify the investigated properties. In order to conduct the verification, we must take into account that the base model is a kP

system and the properties must be verified in the context of a P system state, after the maximally parallel application of the rules in each compartment, and not in every state of the Promela model. In order to accomplish this, a special boolean variable, called *isPSystemStep*, is used for identifying P system states. This variable is included in the LTL formulae in order to instruct Spin to consider only states where the configuration reaches a relevant point in the system execution.

The following investigations have been conducted using an initial configuration of *noCyo* = 200, *noCyd* = 200, *noDim* = 150, *noMon* = 1, *noBdim* = 35, *noBSites* = 35, and the number of oxygen molecules varying between 100, 200 and 300, aiming to verify the relation between the number of different molecules and the evolution of the system after some key points in the reaction process:

- $F(isPSystemStep \text{ and } nrOxy=0)$  – using an initial configuration of *noOxy* = 100, after 500 steps, *noOxy* will eventually decrease to 0.
- $G((isPSystemStep \text{ and } noMon \leq 2 * noDim) \text{ or } !isPSystemStep)$  – with initial *noOxy* = 300, after 500 steps, the number of *Fnr* monomers will be at most half the number of *Fnr* dimers.
- $G(!(noOxy=0 \text{ and } prevNoOxy = 0) \text{ or } noMon \leq prevNoMon \text{ or } !isPSystemStep)$  – after the point when *noOxy*=0, no *Fnr* monomers will be produced, with an initial value of *nrOxy*= 100 and 200 execution steps.
- $G(!(noOxy=0 \text{ and } prevNoOxy=0) \text{ or } noBSites \leq prevNoBSites \text{ or } !isPSystemStep)$  – with an initial configuration of *nrOxy*=100 and 200 execution steps, no more binding sites will become available after the point when *noOxy*=0.
- $G(isPSystemStep \text{ or } !(noOxy=0 \text{ and } prevNoOxy=0 \text{ and } noBSites=0 \text{ and } prevNoBSites=0) \text{ or } noBDim=prevNoBDim)$  – after reaching the state when *noOxy*=0 and *noBSites*=0, the number of bounded dimer molecules will remain unchanged, for an initial *noOxy* = 200, and 1,000 execution steps.
- $G(!isPSystemStep \text{ or } !(noOxy= 0 \text{ and } prevNoOxy=0) \text{ or } noBDim \geq prevNoBDim)$  – using initial 100 oxygen molecules, after 200 steps, the number of bounded dimer molecules will remain the same or at most increase, after reaching the state when *noOxy* = 0.
- $G(!isPSystemStep \text{ or } (noOxy = 0 \rightarrow F(noBDim = 70 \text{ and } isPSystemStep)))$  – for the given configuration, all the existent binding sites will eventually become occupied, after the point when all the oxygen will be consumed, for initial *noOxy* = 100 and 1,000 execution steps.

For the set of properties for which a counterexample was issued, the verification time was up to 3 minutes. On the other hand, for the remaining properties, the time varied between 30 and 40 minutes, depending on the number of iterations and the complexity of the property.

The previous results were obtained by running the models on an Intel Xeon CPU with a speed of 2.4 GHz and 8 GB of RAM.

### 8.6.5 Event-B Versus Promela

As detailed in the previous subsections, the conducted simulations and verifications were supported by modelling the *E. coli* processes into Event-B and Promela formal specification languages and taking advantage of their mature tool support, Rodin and Spin, respectively. Despite both providing powerful modelling capabilities, the two languages are basically very different in their modelling approaches. Event-B models are abstract state machines in which transitions between states are implemented as *events*. An event is a state transition which is specified in terms of *guards* and *actions*. Guards are necessary conditions for an event to be enabled. Actions describe how the occurrence of an event modify some of the variables of the model. On the other hand, Promela provides a powerful set of instructions for describing *concurrent processes* and *inter-process communications*.

Each model takes advantage of its corresponding language constructs for implementing the necessary functionality. Having a formalism based on the set theory, the Event-B model uses functions, sets and set operators as building blocks for specifying the molecule evolution rules. The Promela model implements the corresponding functionality as a scheduling process for synchronizing and running a maximum number iterations for evolving the *E. coli* molecules starting from a given initial configuration. The non-deterministic conditional and cycling instructions available in Promela recommends it as a suitable specification language for modelling the non-deterministic behavior of the different molecules.

Despite the fact that both languages proved suitable modelling capabilities for in-silico *E. coli* simulations, the verification and simulation tasks performed better in case of Spin in terms of complexity of the properties being verified, memory and time, recommending it once again and increasing the confidence for being a leading model checking tool in its class.

## 8.7 Conclusions

A constant concern of biologists, the bacteria *E. coli* has an interesting behavior in relation to the level of oxygen in the environment. In this chapter, we built a (simplified) kernel P system model based on the FLAME model of *E. coli* and we complemented simulations with formal verifications that can check various patterns of behaviour on the model. For instance, such verification can provide “sanity checks” on the model, which should increase the confidence of modellers and biologists that the models behaves as expected. Then, the kernel P system was implemented in two modelling languages, Event-B and Promela, and simulation results show that the kP system is consistent with the original FLAME model. Using the two implementations and the associated model checkers, several properties, formulated using the LTL formalism, were verified. Event-B proved to be more convenient for modelling, while

Spin was more efficient for simulation and verification. The models and results used in this chapter are uploaded on the web page of the MuVeT project [2].

In order to complement our approach, we are currently investigating invariant property generation (using a tool called Daikon), which should help biologists to build a good list of properties to be verified on the model, thus increasing the quality of the used model. Our future work will also concentrate on finding strategies to verify more complicated properties using the model checkers, developing the model by adding the *ArcBA* molecules, and applying similar methodology to other biological entities.

**Acknowledgments** This work was partially supported by the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, via MuVeT project [2], code PN-II-ID-PCE-2011-3-0688

## References

1. FLAME web site. <http://flame.ac.uk>
2. MuVeT web site. <http://muvet.ifsoft.ro/e-coli.html>
3. ProB web site. <http://www.stups.uni-duesseldorf.de/ProB>
4. Spin web site. <http://spinroot.com>
5. SUMO project. <http://sysmo-sumo.mpi-magdeburg.mpg.de/trac/wiki/public>
6. J.-R. Abrial, *Modeling in Event-B. System and software engineering*. (Cambridge University Press, New York, 2010)
7. J.-R. Abrial, S. Hallerstede, Refinement, decomposition, and instantiation of discrete models: Application to Event-B. *Fundam. Informaticae* **77**, 1–28 (2007)
8. O. Andrei, M. Calder, Trend-based analysis of a population model of the AKAP scaffold protein. *Trans. Comput. Syst. Biol.* **7625**, 1–25 (2012)
9. O. Andrei, G. Ciobanu, D. Lucanu, A rewriting logic framework for operational semantics of membrane systems. *Theor. Comput. Sci.* **373**(3), 163–181 (2007)
10. D. Benque, S. Bourton, C. Cockerton, B. Cook, J. Fisher, S. Ishtiaq, N. Piterman, A. Taylor, M. Vardi, in *Proceedings of CAV'12*, BMA: visual tool for modeling and analyzing biological networks. vol. 7358 of LNCS (Springer, 2012), pp. 686–692
11. F. Bernardini, M. Gheorghe, F.J. Romero-Campero, N. Walkinshaw, in *WMC 2007*, A hybrid approach to modeling biological systems, vol. 4860 of LNCS (Springer, 2007), pp. 138–159
12. J. Blakes, J. Twycross, F.J. Romero-Campero, N. Krasnogor, The Infobiotics workbench: an integrated in silico modelling platform for systems and synthetic biology. *Bioinformatics* **27**(23), 3323–3324 (2011)
13. G. Ciobanu, M.J. Pérez-Jiménez, G. Păun, (eds.), *Applications of Membrane Computing. Natural Computing Series* (Springer, Heidelberg, 2006)
14. E. Csuhaj-Varjú, M. Gheorghe, M. Stannett, in *Proceedings of UCNC'12*. P systems controlled by general topologies, vol. 7445 of LNCS (Springer, 2012), pp. 70–81
15. D. Díaz-Pernil, C. Graciani, M. Gutierrez-Naranjo, I. Pérez-Hurtado, M. Pérez-Jiménez. Software for P systems, in *The Oxford Handbook of Membrane Computing*, ed. by Gh. Păun, G. Rozenberg, A. Salomaa (Oxford University Press, Oxford, 2010), pp. 118–143
16. D. Díaz-Pernil, I. Pérez-Hurtado, M. Pérez-Jiménez, A. Riscos-Núñez, in *Proceedings of WMC'08*. A P-Lingua programming environment for membrane computing, vol. 5391 of LNCS (Springer, Heidelberg, 2008), pp. 187–203
17. S. Eilenberg, *Automata, Languages and Machines* (Academic Press, New York, 1994)
18. M. Gheorghe, F. Ipate, C. Dragomir. A kernel P system, in *Proceedings of BWMC10*. Fénix Editora, Seville, 2012, pp. 153–170



19. M. Gheorghe, F. Ipate, R. Lefticaru, C. Dragomir, in *Proceedings of CMC'10*, An integrated approach to P systems formal verification, (ProBusiness Verlag, Heidelberg, 2010), pp. 225–238
20. M. Gheorghe, F. Ipate, R. Lefticaru, M.J. Pérez-Jiménez, A. Turcanu, L. Mierla, L. Valencia Cabrera, F.M. Garcia-Quismondo, 3-Col problem modelling using simple kernel P systems. *Int. J. Comput. Math.* **90**(4), 816–830 (2013)
21. A. Hinton, M. Z. Kwiatkowska, G. Norman, D. Parker, in *Proceedings of TACAS'06*. PRISM: a tool for automatic verification of probabilistic systems, vol. 3920 of LNCS (Springer, Heidelberg, 2006), pp. 441–444
22. M. Holcombe, X-machines as a basis for dynamic system specification. *Softw. Eng. J.* **3**(2), 69–76 (1988)
23. M. Holcombe et al., Modelling complex biological systems using an agent-based approach. *Integr. Biol.* **4**, 53–64 (2012)
24. G. Holzmann, The model checker SPIN. *IEEE Trans. Softw. Eng.* **5**(23), 279–295 (1997)
25. F. Ipate, T. Balanescu, P. Kefalas, M. Holcombe, G. Eleftherakis, A new model of communicating stream X-machine systems. *Rom. J. Inf. Sci. Technol.* **6**(1–2), 165–184 (2003)
26. F. Ipate, R. Lefticaru, L. Mierla, L. Valencia Cabrera, H. Han, G. Zhang, C. Dragomir, M. J. Pérez-Jiménez, M. Gheorghe, in *Proceedings of BIC-TA'13*. Kernel P systems: applications and implementations, vol. 202 of Advances in Intelligent Systems and Computing (Springer, 2013), pp. 1081–1089
27. F. Ipate, R. Lefticaru, C. Tudose, Formal verification of P systems using SPIN. *Int. J. Found Comput. Sci.* **22**(1), 133–142 (2011)
28. F. Ipate, A. Turcanu, in *Proceedings of BWMC9*. Modelling, verification and testing of P systems using Rodin and ProB, (Fénix Editora, Seville, 2011), pp. 209–220
29. M. Kwiatkowska, G. Norman, D. Parker. Symbolic Systems Biology, chapter Probabilistic Model Checking for Systems Biology. (Jones and Bartlett, 2010), pp. 31–59
30. M. Leuschel, M. Butler, ProB: an automated analysis toolset for the B method. *Int. J. Softw. Tools for Technol. Transf.* **10**(2), 185–203 (2008)
31. A. Obtulowicz, G. Paun, (In search of) Probabilistic P systems. *Biosystems* **70**(2), 107–121 (2003)
32. I. Pérez-Hurtado, L. V. Cabrera, M. J. Pérez-Jiménez, M. A. Colomer, in *Proceedings of BIC-TA'10*. MeCoSim: a general purpose software tool for simulating biological phenomena by means of P systems, (IEEE Xplore, 2010), pp. 637–643
33. G. Păun, Computing with membranes. *J. Comput. Sys. Sci.* **61**(1), 108–143 (2000)
34. G. Păun, G. Rozenberg, A. Salomaa (eds.), *The Oxford Handbook of Membrane Computing* (Oxford University Press, Oxford, 2010)
35. P. Richmond, D. Walker, S. Coakley, D. Romano, High performance cellular level agent-based simulation with FLAME for the GPU. *Briefings in Bioinf.* **11**(3), 334–347 (2010)
36. A. Turcanu, F. Ipate, in *Proceedings of CMC'11*. Modelling, testing and verification of P systems with active membranes using Rodin and ProB, (Paris-Est University Press, Paris, 2011), pp. 459–468