Muffy Calder
Stephen Gilmore (Eds.)

LNBI 4695

Computational Methods in Systems Biology

International Conference CMSB 2007
Edinburgh, Scotland, September 2007
Proceedings

Springer

Lecture Notes in Bioinformatics 4695

Subseries of Lecture Notes in Computer Science

Muffy Calder Stephen Gilmore (Eds.)

Computational Methods in Systems Biology

International Conference CMSB 2007
Edinburgh, Scotland, September 20-21, 2007
Proceedings

Springer

Volume Editors

Muffy Calder
Department of Computing Science
The University of Glasgow
Glasgow, Scotland
E-mail: muffy@dcs.gla.ac.uk

Stephen Gilmore
Laboratory for Foundations of Computer Science
The University of Edinburgh
Edinburgh, Scotland
E-mail: stg@inf.ed.ac.uk

Library of Congress Control Number: 2007934798

CR Subject Classification (1998): I.6, D.2.4, J.3, H.2.8, F.1.1

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN 0302-9743
ISBN-10 3-540-75139-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-75139-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12162642 06/3180 5 4 3 2 1 0

Preface

Systems biology is an exciting new field bringing together life scientists, mathematicians, computer scientists and engineers to explore a new and deeper understanding of biological systems. Computational models and methods of analysis are essential components of the systems biology programme, not only reflecting, but also driving wet lab experimentation and the formation of new hypotheses about system behaviour.

This volume contains the proceedings of the fifth meeting of the international conference on Computational Methods in Systems Biology. The first conference was in Trento, Italy in 2003. The second meeting was in Paris in 2004, and in 2005 the conference came to Edinburgh for the first time. Last year's meeting was again in Trento and this year the conference was again in Edinburgh.

This year the conference attracted over 60 paper submissions. Sixteen of these were selected for presentation at the conference. In choosing the 16 best papers, the conference Chairs received wonderful support from the Programme Committee, who delivered thorough and insightful reviews of all papers in a very short time scale. We thank all of the members of the Programme Committee and their sub-referees for their industriousness. We also thank the authors for responding swiftly to the comments of the referees and revising their papers to address these comments earnestly.

The electronic submission of papers, refereeing and Programme Committee work were made possible by the excellent EasyChair free conference management system. EasyChair managed all of the aspects of the review process from submission to review and discussion, through to sending decisions by e-mail to authors. EasyChair compiled the list of referees which appears in this front matter. We give hearty thanks to Andrei Voronkov for providing this wonderful service to the scientific community.

The conference received financial support this year from the e-Science Institute, the Centre for Systems Biology in Edinburgh, and Microsoft Research, Cambridge. In addition, the Engineering and Physical Sciences Research Council supported the conference and contributed to the student bursaries, which we distributed to PhD students to allow them to attend the conference free of charge.

The conference this year was held in the e-Science Institute, Edinburgh. Lee Callaghan and the administrative team at the e-Science Institute provided excellent support for all of the organisational aspects of the conference, allowing us to concentrate on the technical aspects. We received additional support from the administrative staff in our respective departments, assisting with the preparation of this volume, and planning the associated opening reception and conference dinner.

We were very fortunate this year to have two outstanding invited speakers in Daniel T. Gillespie and Mark Girolami.

July 2007

Muffy Calder
Stephen Gilmore

Organization

The organisers and Co-chairs of the CMSB 2007 conference are Muffy Calder of the University of Glasgow and Stephen Gilmore of the University of Edinburgh.

Steering Committee

Finn Drabløs, Norwegian University of Science and Technology, Trondheim (Norway)
Monika Heiner, TU Cottbus (Germany)
Patrick Lincoln, Stanford Research International (USA)
Satoru Miyano, University of Tokyo (Japan)
Gordon Plotkin, University of Edinburgh (UK)
Corrado Priami, The Microsoft Research – University of Trento Centre for Computational and Systems Biology (Italy)
Magali Roux-Rouquié, CNRS-UPMC (France)
Vincent Schachter, Genoscope, Evry (France)
Adelinde Uhrmacher, University of Rostock (Germany)

Programme Committee

Alexander Bockmayr, Freie Universität Berlin (Germany)
Muffy Calder (Co-chair), University of Glasgow (UK)
Luca Cardelli, Microsoft Research Cambridge (UK)
Vincent Danos, CNRS, Université Denis Diderot (France)
Pierpaolo Degano, Università di Pisa (Italy)
Finn Drabløs, Norwegian University of Science and Technology, Trondheim (Norway)
François Fages, INRIA Rocquencourt (France)
Anthony Finkelstein, University College London (UK)
Stephen Gilmore (Co-chair), University of Edinburgh (UK)
David Harel, Weizmann Institute (Israel)
Monika Heiner, TU Cottbus (Germany)
Walter Kolch, Beatson Institute for Cancer Research (UK)
Ina Koch, Technische Fachhochschule Berlin (Germany)
Gethin Norman, University of Birmingham (UK)
Corrado Priami, The Microsoft Research – University of Trento Centre for Computational and Systems Biology (Italy)
Stephen Ramsey, Institute for Systems Biology, Seattle (USA)
Adelinde Uhrmacher, University of Rostock (Germany)

Referees

Joerg Ackermann
Paolo Ballarini
Roberto Barbuti
Maurice ter Beek
Andrea Bracciali
Linda Brodo
Muffy Calder
Enrico Cataldo
Matteo Cavaliere
Davide Chiarugi
Federica Ciocchetta
Pierpaolo Degano
Lorenzo Dematte
Ross Duncan
Jerome Feret
Anthony Finkelstein
Stephen Gilmore
Maria Luisa Guerriero
Jane Hillston
Bjorn Junker
Michal Kiwkowitz
Gunnar W. Klau
Sandy Klemm
Walter Kolch
Sriram Krishnamachari
Jean Krivine
Abdelhalim Larhlimi
Paola Lecca
Sebastian Lehrack
Roberto Maranngoni
Vikas Marda
Radu Mardare
Thierry Martinez
Patrick May
Paolo Milazzo
Orianne Mozemondet
Ivan Mura
David Parker
Dusko Pavlovic
Nadia Pisanti
Razvan Popescu
Heike Pospisil
Davide Prandi
Corrado Priami
Stephen Ramsey
Ronny Richter
Aurélien Rizk
Alessandro Romanel
Mehrnoosh Sadrzadeh
Yvonne Schmitz
Martin Schwarick
Heike Siebert
Sylvain Soliman
Lin Uhrmacher
Vlad Vyshemirsky
Malcolm Walkinshaw

Table of Contents

Chemical Master Equation and Langevin Regimes for a Gene Transcription Model

Raya Khanin[1] and Desmond J. Higham[2]

[1] University of Glasgow, Glasgow, G12 8QQ , UK
[2] University of Strathclyde, Glasgow, G1 1XH, UK

Abstract. Gene transcription models must take account of intrinsic stochasticity. The Chemical Master Equation framework is based on modelling assumptions that are highly appropriate for this context, and the Stochastic Simulation Algorithm (also known as Gillespie's algorithm) allows for practical simulations to be performed. However, for large networks and/or fast reactions, such computations can be prohibitatively expensive. The Chemical Langevin regime replaces the massive ordinary differential equation system with a small stochastic differential equation system that is more amenable to computation. Although the transition from Chemical Master Equation to Chemical Langevin Equation can be heuristically justified, there is very little guidance available about how closely the two models match. Here, we consider a transcription model from the recent literature and show that it is possible to compare first and second moments in the two stochastic settings. To analyse the Chemical Master Equation we use some recent work of Gadgil, Lee and Othmer, and to analyse the Chemical Langevin Equation we use Ito's Lemma. We find that there is a perfect match—both modelling regimes give the same means, variances and correlations for all components in the system. The model that we analyse involves 'unimolecular reactions', and we finish with some numerical simulations involving dimerization to show that the means and variances in the two regimes can also be close when more general 'bimolecular reactions' are involved.

1 Background

Several experimental techniques are now available to measure gene expression, even at the single cell level [1,2,3]. In parallel, mathematical models and simulation algorithms have been developed to explain these observations and make new predictions [4,5,6,7,8,9,10]. Key modeling and simulation challenges in this area are that (a) some components may be present in relatively small quantities, (b) there can be a wide range of natural time scales in operation, and (c) on the level at which observations are made, the process is inherently stochastic. A Markov process, or *Chemical Master Equation* (CME) framework is highly appropriate in this context, and is now widely used. The CME methodology and an accompanying simulation algorithm can be traced back to the work of Gillespie in the chemical kinetics literature [11,12]. Recent overviews can be found

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 1–14, 2007.

in [6,13,14] and we note that there are close connections to Petri nets, discrete event simulation and birth-and-death processes [15].

Because the CME framework takes account of every reaction, for many realistic models it is too computationally expensive to be useful. The *Chemical Langevin Equation* (CLE) provides an alternative model that retains some of the main features of the CME whilst making simulations more feasible. The CLE, which takes the form of an Ito stochastic differential equation (SDE), can be derived from the CME via a series of reasonable modeling assumptions [16], and under the extreme case where fluctuations in the CLE are ignored, we recover the traditional deterministic *Reaction Rate Equation* (or *Law of Mass Action*). Many authors are now developing *multi-scale* simulation methods that automatically operate in the cheapest modeling regime that captures the appropriate behaviour [17,18]. For this reason it is important to have an understanding of how the different modelling regimes compare. This motivates the work here, where the means and variances of the CME and CLE are compared for a recent gene transcription model. To analyse the CME we make use of the general first-order reaction theory of Gadgil et al. [19] and to analyse the CLE we perform what appears to be the first application of Ito's lemma in this context.

The article is organised as follows. In the next section we give a very simple example that illustrates the main concepts involved in our work. Then in section 3 we set up the general specification of the CME and CLE and introduce Ito's lemma. The gene regulation model is described in section 4 and moments for the CME and CLE are derived analytically in sections 5 and 6 respectively. A numerical experiment involving dimerization is given in section 7 to show that similar behaviour can also arise when we leave the first-order realm.

2 Illustrative Example: Unimolecular Decay

To illustrate the ideas in this work, we begin with the simplest possible type of reaction; unimolecular decay. We suppose that there is only one species, S, in our system, and the only event that can take place at any time is that one molecule of S may decay. We could write the system symbolically as

$$S \xrightarrow{c} \emptyset \tag{1}$$

Here, $c > 0$ is a constant that relates to the propensity of the decay process.

We suppose that initially, at time $t = 0$, the number of molecules of S is known to be N. The state of the system at time t is fully described by a non-negative integer $X(t)$, representing the number of molecules of S present. So $X(t)$ may take any of the values $N, N-1, N-2, \ldots, 1, 0$. In the CME setting we regard $X(t)$ as a discrete-valued random variable at each point in time, and work in terms of the probability $p_i(t)$ that $X(t) = i$, arriving at the ordinary differential equation (ODE) system

$$\frac{d}{dt}p_N(t) = -cNp_N(t), \tag{2}$$

$$\frac{d}{dt}p_i(t) = c \cdot (i+1) \cdot p_{i+1}(t) - c \cdot i \cdot p_i(t), \quad \text{for } i = N-1, N-2, \ldots, 0. \tag{3}$$

The general ODE (3) has a natural interpretation. The rate of change of $p_i(t)$ has a positive contribution $c \cdot (i+1) \cdot p_{i+1}(t)$, which corresponds to the fact that we arrive at state i via one decay from state $i+1$. Conversely, there is a negative contribution $-c \cdot i \cdot p_i(t)$ due to the fact that, when in state i, we leave that state when a decay takes place.

The system (2)–(3) is readily solved to give

$$p_i(t) = \frac{N!}{i!(N-i)!} e^{-cit} \left(1 - e^{-ct}\right)^{N-i}, \quad \text{for } i = 0, 1, 2, \ldots, N. \tag{4}$$

Using $\mathbb{E}[\cdot]$ and $\mathsf{Var}[\cdot]$ to denote the mean and variance, respectively, it follows that

$$\mathbb{E}\left[X(t)\right] = Ne^{-ct} \quad \text{and} \quad \mathsf{Var}\left[X(t)\right] = Ne^{-ct}\left(1 - e^{-ct}\right). \tag{5}$$

Details can be found, for example, in [20] by observing that this system corresponds to a pure death process in the context of stochastic population modelling.

In the CLE setting, we represent the amount of species S present at time t by the real-valued stochastic process $Y(t)$. In other words, at each time t, $Y(t)$ is a continuous-valued random variable. The CLE is then the Ito SDE [21,22]

$$dY(t) = -cY(t)\,dt - \sqrt{cY(t)}\,dW(t), \quad Y(0) = N. \tag{6}$$

Because the drift coefficient $-cY(t)$ is linear, it follows immediately that $\mathbb{E}\left[Y(t)\right]$ satisfies the ODE that arises when the noise is switched off, giving

$$\mathbb{E}\left[Y(t)\right] = Ne^{-ct}. \tag{7}$$

To find the second moment, we may apply Ito's lemma, as described in section 3.2, to get

$$\frac{d}{dt}\mathbb{E}\left[Y(t)^2\right] = -2\,c\,\mathbb{E}\left[Y(t)^2\right] + c\,\mathbb{E}\left[Y(t)\right].$$

Using the expression (7), this solves to give $\mathbb{E}\left[Y(t)^2\right] = Ne^{-ct}$, so that

$$\mathsf{Var}\left[Y(t)\right] = Ne^{-ct}\left(1 - e^{-ct}\right). \tag{8}$$

Comparing (7) and (8) with (5), we see that the models give precisely the same expressions for the mean and variance of S. This happens despite the fact that one uses the discrete, integer-valued state vector $X(t)$ and the other uses the real-valued $Y(t)$.

For completeness, we mention that the law of mass action, or reaction rate equation, formulation for the system (1) has the form of a scalar ODE $dz(t)/dt = -cz(t)$, where $z(t)$ is a deterministic real-valued quantity representing the amount of S present at time t. This is precisely the ODE for the mean in the CLE, and hence $z(t) = \mathbb{E}[Y(t)] = Ne^{-ct}$.

Two features of the CLE (6) for this simple model are generic.

1 The diffusion coefficient is nonlinear.
2 The description of the problem involves a square root, and hence the problem is only well defined if the solution remains non-negative.

With regard to the second point, the particular CLE (6) is a special case of a *square root process*. These SDEs are widely used as interest rate models in mathematical finance, and it can be shown that the solution in (6) maintains non-negativity with probability one [22]. However, we note that the issue of negative solutions seems to be open for general CLEs. In this work, we will always assume that the CLE has a well-defined, unique solution.

The main result in this article is that the coincidence of CME and CLE mean and variance in the simple model (1) carries through to a gene transcription model.

3 Stoichiometric Formalization

3.1 Chemical Master Equation

Suppose that there are N chemical species, $S_1, S_2, \ldots, S_N$ taking part in M different chemical reactions. In the CME formulation, we have a state vector $\mathbf{X}(t) \in \mathbb{R}^N$ whose ith component, $X_i(t)$, denotes the number of molecules of S_i present at time t. Hence, each $X_i(t)$ is a non-negative integer. For each $1 \leq j \leq M$ we have a *stoichiometric vector* $\boldsymbol{\nu}_j \in \mathbb{R}^N$, and *propensity function* $a_j(\mathbf{X}(t))$, such that the jth reaction takes place over the infinitesimal interval $[t, t+dt)$ with probability $a_j(\mathbf{X}(t))\,dt$ and causes the change $\mathbf{X}(t) \mapsto \mathbf{X}(t) + \boldsymbol{\nu}_j$ to the state vector.

Letting $P(\mathbf{x}, t)$ denote the probability that $\mathbf{X}(t) = \mathbf{x}$, the CME is the ODE system

$$\frac{dP(\mathbf{x},t)}{dt} = \sum_{j=1}^{M} \left(a_j(\mathbf{x}-\boldsymbol{\nu}_j)P(\mathbf{x}-\boldsymbol{\nu}_j,t) - a_j(\mathbf{x})P(\mathbf{x},t)\right). \tag{9}$$

Generally, the CME cannot be solved analytically in any useful way, although Gillespie's *Stochastic Simulation Algorithm* (SSA) [11,12] gives a way to compute realisations of $\{t, \mathbf{X}(t)\}$ that respect the CME. However, in the case where all reactions are unimolecular (or first-order), detailed analysis is possible, both for the first and second moments [19] and the general distributions [23]. In this work we will show that, at least for a specific gene regulation model, useful analytical results can also be derived for the CLE formulation described in the next subsection.

3.2 Chemical Langevin Equation

The CLE uses a real-valued random variable $\mathbf{Y}(t) \in \mathbb{R}^N$ to describe the state of the system at time t. The jth component $Y_j(t)$ represents the amount of species j. In moving from the CME to the CLE we (typically) make a dramatic

reduction in the number of components, but pay the price that each component is a real-valued random variable, rather than a non-negative integer. The CLE takes the form of an Ito SDE [21,22]

$$d\,\mathbf{Y}(t) = \sum_{j=1}^{M} \boldsymbol{\nu}_j\, a_j(\mathbf{Y}(t))\, dt + \sum_{j=1}^{M} \boldsymbol{\nu}_j \sqrt{a_j(\mathbf{Y}(t))}\, dW_j(t), \tag{10}$$

where the $\{W_j(t)\}_{j=1}^{M}$ are independent Brownian motions.

As background for the SDE analysis in section 6, we now state the relevant part of Ito's lemma; see, for example, [22]. For the general Ito SDE system with n components and d independent Brownian motions

$$dY_i(t) = b_i\left(\mathbf{Y}(t)\right) dt + \sum_{j=1}^{d} \sigma_{ij}\left(\mathbf{Y}(t)\right) dW_j(t), \quad 1 \le i \le n, \tag{11}$$

we let

$$a\left(\mathbf{Y}(t)\right) := \boldsymbol{\sigma}\left(\mathbf{Y}(t)\right) \boldsymbol{\sigma}\left(\mathbf{Y}(t)\right)^T \in \mathbb{R}^{n\times n}. \tag{12}$$

Then for any function $f : \mathbb{R}^n \to \mathbb{R}$ that is twice continuously differentiable, Ito's lemma tells us that

$$\begin{aligned} d\, f\left(\mathbf{Y}(t)\right) = &\left(\sum_{i=1}^{n} \frac{\partial f\left(\mathbf{Y}(t)\right)}{\partial x_i} b_i\left(\mathbf{Y}(t)\right) + \tfrac{1}{2} \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 f(\mathbf{Y}(t))}{\partial x_i \partial x_j} a_{ij}\left(\mathbf{Y}(t)\right) \right) dt \\ &+ \text{mart.}, \end{aligned} \tag{13}$$

where "mart." denotes a martingale whose precise form is not relevant to our work. We will use two particular cases of f. When $f\left(\mathbf{Y}\right) = Y_k^2$, (13) becomes

$$d\left(Y_k^2\right) = \left(2\, Y_k\, b_k\left(\mathbf{Y}(t)\right) + a_{kk}\left(\mathbf{Y}(t)\right)\right) dt + \text{mart.} \tag{14}$$

and when $f\left(\mathbf{Y}\right) = Y_k Y_l$, for $k \neq l$, it becomes

$$d\left(Y_k Y_l\right) = \left(Y_l b_k\left(\mathbf{Y}(t)\right) + Y_k b_l\left(\mathbf{Y}(t)\right) + \tfrac{1}{2} a_{kl}\left(\mathbf{Y}(t)\right) + \tfrac{1}{2} a_{lk}\left(\mathbf{Y}(t)\right)\right) dt + \text{mart.} \tag{15}$$

4 Gene Regulation Model

We now consider a model of eukaryotic gene regulation, originally proposed in [24]. This model incorporates two states of promoters: an inactive state, D, not permissive of transcription, and an active state $D^\star$ that is competent for transcription. Transition between the two states of promoter is reversible and the total number of promoters is conserved, i.e. $D + D^\star = D_T$. Transcription takes place from the active state $D^\star$ with the linear rate k_r, resulting in production of messenger RNA (mRNA) molecules that decay with rate γ_r. Proteins P are translated from mRNA molecules with linear rate k_p and they decay with rate γ_p.

This model of gene regulation could be described by the following reactions:

$$D \underset{k_d}{\overset{k_a}{\rightleftharpoons}} D^\star \tag{16}$$

$$D^\star \xrightarrow{k_r} M + D^\star \tag{17}$$

$$M \xrightarrow{\gamma_r} \emptyset \tag{18}$$

$$M \xrightarrow{k_p} P + M \tag{19}$$

$$P \xrightarrow{\gamma_p} \emptyset \tag{20}$$

We note that the representation in (16)–(20) leaves room for some ambiguity. A formal and complete specification of the system in terms of stoichiometric vectors and propensity functions can be found in section 6.

It is tempting to reduce the first two reactions (16) that involve two species D and $D^\star$ to just one reaction involving $D^\star$ by exploiting the constraint $D + D^\star = D_T$. We could argue that $D^\star$ is produced with the rate $k_a D_T$ and decays with the rate $(k_a + k_d)D^\star$:

$$\begin{array}{ccc} & \overset{k_a D_T}{} & \\ \emptyset & \longrightarrow & D^\star \\ & \underset{k_a + k_d}{} & \\ D^\star & \longrightarrow & \emptyset \end{array}$$

In this formalization, however, we cannot guarantee that once $D^\star = D_T$ no more production of the active state $D^\star$ will occur. This opens up the possibility of $D^\star > D_T$, which violates the conservation law. Hence, we will work with the full system.

5 Moments for Chemical Master Equation

Gadgil et al. [19] considered generic systems of first-order chemical reactions and derived ODEs that describe the evolution of the first two moments of all species. They split first-order reactions into four categories. The gene transcription model (16)–(20) fits into that framework and involves three of these categories. Reactions $D \rightleftharpoons D^\star$ are of *conversion* type, $D^\star \to M + D^\star$ and $M \to P + M$ are *catalytic*, i.e. the reaction affects one species at a rate that is is proportional to some other species, and $M \to \emptyset$ and $P \to \emptyset$ are *degradation* type reactions. The fourth type of reaction in [19], production from a source, is not present in this model.

If we use the symbols $D(t)$, $D^\star(t)$, $M(t)$ and $P(t)$ to denote the number of molecules of each species present at time t, [19, equation (28)] shows that the mean values arising from the CME model satisfy the ODE system

$$\frac{d}{dt}\begin{bmatrix}\mathbb{E}\left[D(t)\right]\\ \mathbb{E}\left[D^\star(t)\right]\\ \mathbb{E}\left[M(t)\right]\\ \mathbb{E}\left[P(t)\right]\end{bmatrix} = \mathcal{K}\cdot\begin{bmatrix}\mathbb{E}\left[D(t)\right]\\ \mathbb{E}\left[D^\star(t)\right]\\ \mathbb{E}\left[M(t)\right]\\ \mathbb{E}\left[P(t)\right]\end{bmatrix}, \quad \text{where } \mathcal{K} = \begin{bmatrix}-k_a & k_d & 0 & 0\\ k_a & -k_d & 0 & 0\\ 0 & k_r & -\gamma_r & 0\\ 0 & 0 & k_p & -\gamma_p\end{bmatrix}. \tag{21}$$

Then introducing a time dependent symmetric matrix $V(t) \in \mathbb{R}^{4\times 4}$ to store the second moments and correlations in the form

$$V(t) := \begin{bmatrix}\mathbb{E}\left[D(t)^2 - D(t)\right] & \mathbb{E}\left[D(t)D^\star(t)\right] & \mathbb{E}\left[D(t)M(t)\right] & \mathbb{E}\left[D(t)P(t)\right]\\ \mathbb{E}\left[D(t)D^\star(t)D(t)\right] & \mathbb{E}\left[D^\star(t)^2 - D^\star(t)\right] & \mathbb{E}\left[M(t)D^\star(t)\right] & \mathbb{E}\left[P(t)D^\star(t)\right]\\ \mathbb{E}\left[M(t)D(t)\right] & \mathbb{E}\left[M(t)D^\star(t)\right] & \mathbb{E}\left[M(t)^2 - M(t)\right] & \mathbb{E}\left[M(t)P(t)\right]\\ \mathbb{E}\left[P(t)D(t)\right] & \mathbb{E}\left[P(t)D^\star(t)\right] & \mathbb{E}\left[M(t)P(t)\right] & \mathbb{E}\left[P(t)^2 - P(t)\right]\end{bmatrix},$$

we may appeal to [19, equation (29)], which says

$$\frac{d}{dt}V(t) = \mathcal{K}V(t) + \left(\mathcal{K}V(t)\right)^T + \Gamma(t) + \Gamma(t)^T, \tag{22}$$

where, in our case, $\Gamma(t) \in \mathbb{R}^{4\times 4}$ has the form

$$\Gamma(t) = \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & k_r\mathbb{E}[D^\star(t)] & 0\\ 0 & 0 & 0 & k_p\mathbb{E}[M(t)]\\ 0 & 0 & 0 & 0\end{bmatrix}.$$

Substituting the expressions in (21) for the means, we obtain ODEs for the second moments;

$$\frac{d\,\mathbb{E}[D^2]}{dt} = -2k_a\mathbb{E}[D^2] + 2k_d\mathbb{E}[DD^\star] + k_a\mathbb{E}[D] + k_d\mathbb{E}[D^\star], \tag{23}$$

$$\frac{d\,\mathbb{E}[D^{\star 2}]}{dt} = -2k_d\mathbb{E}[D^{\star 2}] + 2k_a\mathbb{E}[DD^\star] + k_a\mathbb{E}[D] + k_d\mathbb{E}[D^\star], \tag{24}$$

$$\frac{d\,\mathbb{E}[M^2]}{dt} = -2\gamma_r\mathbb{E}[M^2] + 2k_r\mathbb{E}[D^\star M] + k_r\mathbb{E}[D^\star] + \gamma_r\mathbb{E}[M], \tag{25}$$

$$\frac{d\,\mathbb{E}[P^2]}{dt} = -2\gamma_p\mathbb{E}[P^2] + 2k_p\mathbb{E}[MP] + k_p\mathbb{E}[M] + \gamma_p\mathbb{E}[P]. \tag{26}$$

Here, and henceforth, to avoid cluttering the equations we suppress the time dependence, so, for example, $D(t)$ is written simply as D. Similarly, for the

correlations we find that

$$\frac{d\,\mathbb{E}[DD^\star]}{dt} = -(k_a + k_d)\mathbb{E}[DD^\star] + k_a\mathbb{E}[D^2] + k_d\mathbb{E}[D^{\star 2}] - k_a\mathbb{E}[D] - k_d\mathbb{E}[D^\star] \tag{27}$$

$$\frac{d\,\mathbb{E}[DM]}{dt} = -(k_a + \gamma_r)\mathbb{E}[DM] + k_r\mathbb{E}[DD^\star] + k_d\mathbb{E}[D^\star M], \tag{28}$$

$$\frac{d\,\mathbb{E}[DP]}{dt} = -(k_a + \gamma_p)\mathbb{E}[DP] + k_p\mathbb{E}[DM] + k_d\mathbb{E}[D^\star P], \tag{29}$$

$$\frac{d\,\mathbb{E}[D^\star M]}{dt} = -(k_d + \gamma_r)\mathbb{E}[D^\star M] + k_r\mathbb{E}[D^{\star 2}] + k_a\mathbb{E}[DM], \tag{30}$$

$$\frac{d\,\mathbb{E}[D^\star P]}{dt} = -(k_d + \gamma_p)\mathbb{E}[D^\star P] + k_a\mathbb{E}[DP] + k_p\mathbb{E}[D^\star M], \tag{31}$$

$$\frac{d\,\mathbb{E}[MP]}{dt} = -(\gamma_r + \gamma_p)\mathbb{E}[MP] + k_p\mathbb{E}[M^2] + k_r\mathbb{E}[D^\star P]. \tag{32}$$

6 Moments for Chemical Langevin Equation

The CLE formulation described in subsection 3.2 uses a general state vector $\mathbf{Y}(t)$. For the model (16)–(20), in order to make comparisons easier, we will re-use the notation from section 5, so that

$$\begin{bmatrix} Y_1(t) \\ Y_2(t) \\ Y_3(t) \\ Y_4(t) \end{bmatrix} =: \begin{bmatrix} D \\ D^\star \\ M \\ P \end{bmatrix}.$$

However, we emphasize that D, $D^\star$, M and P in the CLE are real-valued random variables, whereas those in CME take non-negative integer values. We also emphasize that the time-dependency is not made explicit in this notation.

The stoichiometric vectors for reactions (16)–(20) take the form

$$\boldsymbol{\nu}_1 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\nu}_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\nu}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\nu}_4 = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}, \quad \boldsymbol{\nu}_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{\nu}_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix},$$

and the propensity functions are $a_1 = k_a D$, $a_2 = k_d D^\star$, $a_3 = k_r D^\star$, $a_4 = \gamma_r M$, $a_5 = k_p M$ and $a_6 = \gamma_p P$. Hence the CLE (10) is an SDE of the form (11) with drift function $\mathbf{b} : \mathbb{R}^4 \to \mathbb{R}^4$ given by

$$\mathbf{b}(\mathbf{Y}(t)) = \begin{bmatrix} -k_a D + k_d D^\star \\ k_a D - k_d D^\star \\ k_r D^\star - \gamma_r M \\ k_p M - \gamma_p P \end{bmatrix} \tag{33}$$

and diffusion function $\boldsymbol{\sigma} : \mathbb{R}^4 \to \mathbb{R}^{4\times 6}$ given by

$$\boldsymbol{\sigma}(\mathbf{Y}(t)) = \begin{bmatrix} -\sqrt{k_a D} & \sqrt{k_d D^\star} & 0 & 0 & 0 & 0 \\ \sqrt{k_a D} & -\sqrt{k_d D^\star} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{k_r D^\star} & -\sqrt{\gamma_r M} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{k_p M} & -\sqrt{\gamma_p P} \end{bmatrix}. \tag{34}$$

So a in (12) satisfies

$$a = \begin{bmatrix} k_a D + k_d D^\star & -k_a D - k_d D^\star & 0 & 0 \\ -k_a D - k_d D^\star & k_a D + k_d D^\star & 0 & 0 \\ 0 & 0 & k_r D^\star + \gamma_r M & 0 \\ 0 & 0 & 0 & k_p M + \gamma_p P \end{bmatrix}. \tag{35}$$

Because the drift coefficient in (33) is linear, taking expectations in the SDE leads to the linear ODE (21) that we obtained for the CME.

Applying Ito's lemma to $f(\mathbf{Y}) = D^2$, using (14) and the expressions in (33) and (35), we find that

$$d\left(D^2\right) = \left(2D\left(-k_a D + k_d D^\star\right) + \left(k_a D + k_d D^\star\right)\right) dt + \text{mart.}$$

so, after taking expectations,

$$\frac{d\,\mathbb{E}[D^2]}{dt} = -2k_a\mathbb{E}[D^2] + 2k_d\mathbb{E}[DD^\star] + k_a\mathbb{E}[D] + k_d\mathbb{E}[D^\star],$$

which matches (23). Similarly, (14) shows that the other second moments satisfy the ODEs (24)–(26). In the same manner, we may apply Ito's lemma to $f(\mathbf{Y}) = DD^\star$, using (15), to find that

$$\begin{aligned} d\left(DD^\star\right) = &\left(D^\star\left(-k_a D + k_d D^\star\right) + D\left(k_a D - k_d D^\star\right)\right. \\ &\left. + \tfrac{1}{2}\left(-k_a D - k_d D^\star - k_a D - k_d D^\star\right)\right) dt + \text{mart.} \end{aligned}$$

So, after taking expectations,

$$\frac{d\,\mathbb{E}[DD^\star]}{dt} = -(k_a + k_d)\mathbb{E}[DD^\star] + k_a\mathbb{E}[D^2] + k_d\mathbb{E}[D^{\star 2}] - k_a\mathbb{E}[D] - k_d\mathbb{E}[D^\star],$$

matching (27). Similarly, the other correlations are found to satisfy (28)–(32).

In summary, the means, variances and correlations for all components satisfy the same ODEs for both the CME and CLE formulations of the model, and hence they are equal for all time.

7 Numerical Experiment for a Bimolecular Case

All reactions in the model (16)–(20) are first-order in the sense of [19]. An important instance where a first-order model is not sufficient arises when proteins

produced from the mRNA may combine to form complexes, such as dimers. There is ample experimental evidence to suggest that protein subunits can degrade less rapidly when associated in multimeric complexes, an effect referred to in [25] as "cooperative stability". For dimeric transcription factors, this effect leads to a concentration-dependence in the degradation rate because monomers, which are predominant at low concentrations, will be more rapidly degraded. Thus, cooperative stability can effectively widen the accessible range of protein levels in vivo and a few-fold difference between the degradation rate of monomers and dimers can already enhance the function of these circuits substantially. In [25], the effect of cooperative stability through nonlinear degradation in a simple genetic circuit with feedback was studied without incorporating stochastic effects. On the other hand, SSA simulations were used in [26], but without considering rapid degradation of monomers compared to dimers.

To illustrate a model that incorporates dimerization, we begin with a simplified version of the model in section 4 where there is only a single, active, state of the gene

$$\begin{aligned} D^\star &\xrightarrow{k_r} M + D^\star \\ M &\xrightarrow{k_p} P + M \\ M &\xrightarrow{\gamma_r} \emptyset \\ P &\xrightarrow{\gamma_{p_1}} \emptyset \end{aligned}$$

Then we may allow the protein monomers P to form dimers P_2, which degrade less rapidly than the monomers ($\gamma_{p_2} < \gamma_{p_1}$):

$$\begin{aligned} P + P &\underset{k_d}{\overset{k_a}{\rightleftharpoons}} P_2 \\ P_2 &\xrightarrow{\gamma_{p_2}} \emptyset \end{aligned}$$

Our aim is to test whether the correspondence between first and second moments for the CME and CLE that we proved for the model with unimolecular reactions in section 4 carries through to this case, where a dimerization (and hence second-order) reaction is present. A full Monte Carlo simulation of the CME and CLE would be very expensive (for example, the CLE contains seven independent Brownian motions, so an expected value corresponds to an integral over seven dimensions). Hence, we will focus on a reduced model that contains dimerization. If we assume that the protein arises as production from a source and ignore any possible reversibility of the dimerization, we arrive at the computationally simpler model

$$\emptyset \xrightarrow{k_1} P \tag{36}$$

$$P + P \xrightarrow{k_a} P_2 \tag{37}$$

$$P \xrightarrow{\gamma_P} \emptyset \tag{38}$$

$$P_2 \xrightarrow{\gamma_{P_2}} \emptyset \tag{39}$$

We emphasize that we are using this model simply to test whether the conclusions of sections 5–6 are close to holding in an example with second order reactions. Writing the state vector as

$$\begin{bmatrix} X_1(t) \\ X_2(t) \end{bmatrix} =: \begin{bmatrix} P \\ P_2 \end{bmatrix},$$

the stoichiometric vectors take the form

$$\boldsymbol{\nu}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{\nu}_2 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \quad \boldsymbol{\nu}_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \boldsymbol{\nu}_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

and the propensity functions are $a_1 = k_a$, $a_2 = k_a P(P-1)/2$, $a_3 = \gamma_P P$ and $a_4 = \gamma_{P_2} P_2$.

In this case the CLE takes the form

$$d\begin{bmatrix} P \\ P_2 \end{bmatrix} = \begin{bmatrix} k_1 - k_a P(P-1) - \gamma_P P \\ k_a P(P-1)/2 - \gamma_{P_2} P_2 \end{bmatrix} dt + \begin{bmatrix} \sqrt{k_1}\, dW_1 - \sqrt{k_a P(P-1)}\, dW_2 - \sqrt{\gamma_P P}\, dW_3 \\ \sqrt{k_a P(P-1)/2}\, dW_2 - \sqrt{\gamma_{P_2} P_2}\, dW_4 \end{bmatrix}.$$

In this SDE the first equation shows that P is uncoupled from P_2 (a fact which is also clear from the original formulation (36)–(39)), so we may consider separately the SDE

$$dP = (k_1 - k_a P(P-1) - \gamma_P P)\, dt + \sqrt{k_1}\, dW_1 - \sqrt{k_a P(P-1)}\, dW_2 - \sqrt{\gamma_P P}\, dW_3.$$

Taking expectations leads to an ODE for $\mathbb{E}[P]$ that involves $\mathbb{E}[P^2]$. Similarly, applying Ito's lemma to $f(Y) = P^2$ gives an ODE for $\mathbb{E}[P^2]$ that involves $\mathbb{E}[P^3]$. Because the system is not closed, this does not lead to an analytical formula for the moments, and also hints that our moment matching approach from sections 5 and 6 is unlikely to be successful.

We therefore proceed computationally. Choosing the values $k_1 = 5$, $k_a = 0.01$, $\gamma_P = 0.1$ and $\gamma_{P_2} = 0.01$ with initial conditions $P(0) = 10$ and $P_2(0) = 2$, we consider the time interval $0 \le t \le 20$. For the purposes of illustration, in Figure 1 we show one path for the monomer P and the dimer P_2 from the CME, computed with SSA, and from the CLE, approximated with the Euler–Maruyama method [21,22]. We note that the two computations use different, independent, noise sources (from MATLAB's `rand` and `randn`) and hence there is no reason for the

two paths to be close. Then, using Monte Carlo simulations over $K = 10^5$ paths we computed sample mean approximations to $\mathbb{E}[P]$, $\mathbb{E}[P^2]$, $\mathbb{E}[P_2]$ and $\mathbb{E}[P_2^2]$ at time $t = 20$. The results are given in Table 1. Here, we have presented 95% confidence intervals for each sample mean by adding $\pm 1.96\,\mathrm{std}/\sqrt{K}$, where std denotes the sample's standard deviation. All values have been rounded to four significant digits. CLEa denotes the results for Euler–Maruyama using a stepsize $20/500 = 0.04$. The table also shows results for Euler–Maruyama with stepsize 0.004, labeled CLEb, in order to check that numerical discretization errors are not significant.

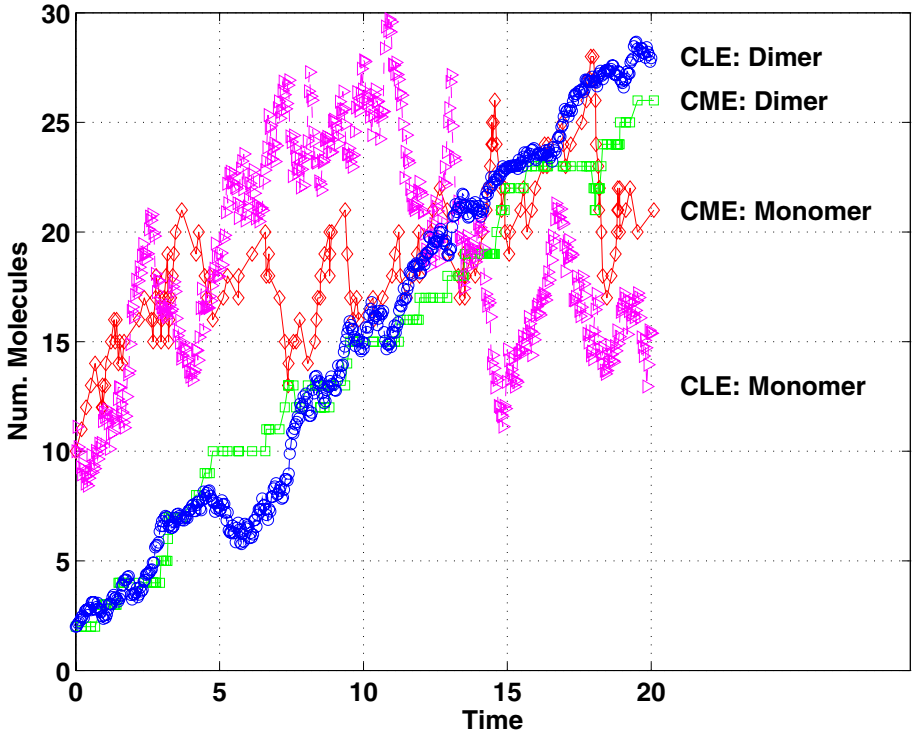


Fig. 1. One path from the Chemical Master Equation (CME) and Chemical Langevin Equation (CME) for the dimerization model (36)–(39). Diamonds: P from CME; Squares: P_2 from CME; Triangles: P from CLE; Circles: P_2 from CLE.

We see from Table 1 that there is overlap between the computed CME and CLE confidence intervals for both sets of first and second moments. We conclude that, to the typical accuracy obtained from large scale Monte Carlo simulations, the means are indistinguishable.

We wrap up by mentioning some possible directions for future work in this area.

Table 1. 95% confidence intervals for Monte Carlo sample mean approximations to $\mathbb{E}[P]$, $\mathbb{E}[P^2]$, $\mathbb{E}[P_2]$ and $\mathbb{E}[P_2^2]$ at time $t = 20$ in (36)–(39) from the CME and CLE. CLEa uses Euler–Maruyama with stepsize 0.04 and CLEb uses Euler–Maruyama with stepsize 0.004.

	$\mathbb{E}[P]$	$\mathbb{E}[P^2]$	$\mathbb{E}[P_2]$	$\mathbb{E}[P_2^2]$
CME	[17.97, 18.01]	[337.3, 339.0]	[28.05, 28.10]	[806.1, 809.2]
CLEa	[17.96, 18.01]	[337.0, 338.8]	[28.07, 28.13]	[807.6, 810.8]
CLEb	[17.97, 18.02]	[337.4, 339.2]	[28.06, 28.11]	[807.0, 810.1]

1. We believe that the moment matching in sections 5 and 6 arises for arbitrary first-order reaction systems, and we are currently putting together a proof of this result.
2. Further computational testing would help to reveal the extent to which moments match for more general models, and in this case some analysis might be possible that gives bounds on the discrepancies and indicates parameter regimes where there is a close match.
3. Generally, there is a need for existence and uniqueness results for the SDEs that can appear in the CLE formulation. The conditions under which the CLE (10) is derived in [16] make it clear that the model is likely to be unrealistic when components $Y_i(t)$ approach zero—this is precisely where the issue of negative arguments inside the square root function rouses itself.

Acknowledgement. This work was sponsored by EPSRC grants GR/S62383/01 and EP/E049370/1 (DJH) and by a Synergy grant from the Universities of Strathclyde and Glasgow (RK and DJH).

References

1. Blake, W., Kaern, M., Cantor, C., Collins, J.: Noise in eukaryotic gene expression. Nature 422(6932), 633–637 (2003)
2. Chubb, J., Trcek, T., Shenoy, S., Singer, R.: Transcriptional pulsing of a developmental gene. Curr. Biol. 16(10), 1018–1025 (2006)
3. Golding, I., Paulsson, J., Zawilski, S., Cox, E.: Real-time kinetics of gene activity in individual bacteria. Cell 123(6), 1025–1036 (2005)
4. Adalsteinsson, D., McMillen, D., Elston, T.C.: Biochemical network stochastic simulator (BioNetS): software for stochastic modeling of biochemical networks. BMC Bioinformatics 5(24) (2004)
5. Crampin, E.J., Schnell, S.: New approaches to modelling and analysis of biochemical reactions, pathways and networks. Progress in Biophysics & Molecular Biology 86, 1–4 (2004)
6. Samad, H.E., Khammash, M., Petzold, L., Gillespie, D.T.: Stochastic modeling of gene regulatory networks. Int. J. Robust and Nonlinear Control 15, 691–711 (2005)
7. Paszek, P.: Modeling stochasticity in gene regulation: characterization in the terms of the underlying distribution function. Bulletin of Mathematical Biology (2007) DOI 10.1007/s11538-006-9176-7

8. Turner, T.E., Schnell, S., Burrage, K.: Stochastic approaches for modelling in vivo reactions. Computational Biology and Chemistry 28, 165–178 (2004)
9. Swain, P.S., Elowitz, M., Siggia, E.D.: Intrinsic and extrinsic contributions to stochasticity in gene expression. Proc. Natl. Acad. Sci. USA 99(20), 12795–12800 (2002)
10. Swain, P.S.: Efficient attenuation of stochasticity in gene expression through post-transcriptional control. J. Mol.Biol. 344(4), 965–976 (2004)
11. Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comp. Phys. 22, 403–434 (1976)
12. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81, 2340–2361 (1977)
13. Gillespie, D.T., Petzold, L.: Numerical simulation for biochemical kinetics. In: Szallasi, Z., Stelling, J., Periwal, V. (eds.) System Modelling in Cellular Biology: From Concepts to Nust and Bolts, pp. 125–147. MIT Press, Cambridge (2006)
14. Higham, D.J.: Modeling and simulating chemical reactions. SIAM Review (to appear)
15. Wilkinson, D.J.: Stochastic Modelling for Systems Biology. Chapman & Hall/CRC (2006)
16. Gillespie, D.T.: The chemical Langevin equation. J. Chem. Phys. 113, 297–306 (2000)
17. Weinan, E., Liu, D., Vanden-Eijnden, E.: Nested stochastic simulation algorithms for chemical kinetic systems with multiple time scales. J. Chem. Phys. 123, 194107 (2005)
18. Cao, Y., Gillespie, D.T., Petzold, L.: The slow-scale stochastic simulation algorithm. J. Chem. Phys. 122, 14116 (2005)
19. Gadgil, C., Lee, C.H., Othmer, H.G.: A stochastic analysis of first-order reaction networks. Bulletin of Mathematical Biology 67, 901–946 (2005)
20. Renshaw, E.: Modelling Biological Populations in Space and Time. Cambridge University Press, Cambridge (1991)
21. Higham, D.J.: An algorithmic introduction to numerical simulation of stochastic differential equations. SIAM Review 43, 525–546 (2001)
22. Mao, X.: Stochastic Differential Equations and Applications. Horwood, Chichester (1997)
23. Jahnke, T., Huisinga, W.: Solving the chemical master equation for monomolecular reaction systems analytically. Journal of Mathematical Biology 54, 1–26 (2007)
24. Raser, J., O'Shea, E.: Control of stochasticity in eukaryotic gene expression. Science 304(5678), 1811–1814 (2004)
25. Buchler, N.E., Gerland, U., Hwa, T.: Nonlinear protein degradation and the function of genetic circuits. Proc. Natl. Acad. Sci. U S A 102, 9559–9564 (2005)
26. Bundschuh, R., Hayot, F., Jayaprakash, C.: The role of dimerization in noise reduction of simple genetic networks. J. Theor. Biol. 220, 261–269 (2003)

Simultaneous Stochastic Simulation of Multiple Perturbations in Biological Network Models

Werner Sandmann

University of Bamberg, Feldkirchenstr. 21, D-96045 Bamberg, Germany
werner.sandmann@wiai.uni-bamberg.de

Abstract. Stochastic models of biological networks are well-established in computational systems biology. However, models are abstractions and parameters may be inaccurate or perturbed. An important topic is thus the sensitivity of analysis results to parameter perturbations. This investigates how seriously potential parameter perturbations affect the analysis results and to which parameters the results are most sensitive. In this paper, a stochastic simulation algorithm is presented that yields results for multiple perturbed models from a single simulation experiment and that is thus able to perform comparisons of results for various parameter sets without explicitly simulating each of these separately. The algorithm essentially makes use of likelihood ratios in a similar fashion as in the Importance Sampling technique for variance reduction. With a suitable adaptation to the context of perturbed model parameters it yields substantial runtime savings compared to multiple separate simulations of the perturbed models without any loss in statistical accuracy.

Keywords: Biological Networks, Stochastic Simulation, Parameter Perturbations, Importance Sampling, Likelihood Ratios.

1 Introduction

Living systems consist of interacting biological components forming networks that determine the functionality of cells and organisms. Gaining insights to such complex living systems is the primary scope of systems biology. Since biological and genetic networks are enormously complex, mathematical modeling and computer based analysis are highly important. Randomness within biological and genetic networks has become apparent and was pointed out many times, and today it is highly evident that stochastic models are capable to appropriately model the dynamics of living systems, see e.g. [1,2,3,4,5,6].

In the stochastic approach, the system state at any time is given by the number of molecules of each species and the system evolves acording to a Markov process where the system dynamics are governed by the chemical master equation which is a system of differential equations. Since direct solution of the chemical master equation is usually analytically intractable due to size and complexity of the underlying chemical reaction sets, stochastic simulation is the most

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 15–31, 2007.

widespread tool for model-based analysis in this context. The most frequently applied method of choice is the direct method for stochastically exact generation of trajectories of the underlying Markov process. In the biological community this is known as the Gillespie algorithm [7,8]. Approximations have been proposed to enhance trajectory generation, most notably the tau-leaping method [9], amongst others.

However, any model is an abstraction, all analysis approaches are applied to an abstract model and the usefulness of the analysis results strongly depends on the appropriateness of the model. Any analysis is at best as good as the model reflects reality. Thus, model parameters must be chosen such that they fit to reality. If they are inaccurate or perturbed, the analysis results are so, too. Highly perturbed results are useless and if not recognized as perturbed, they may lead to wrong conclusions regarding the reality of the system being modeled. Hence, it is important to know how sensitive the results are to parameter perturbations. Small changes in certain model parameters may have a big impact on the results. Common approaches to sensitivity analysis - see [10] for applications to biological models - are usually based on obtaining gradients which imposes a serious additional difficulty that may be even more complex than the model analysis itself. Moreover, the values of gradients do not directly provide results for the considered model but only indicate the degree of sensitivity.

In this paper, an algorithm for simultaneously obtaining trajectories of multiple (perturbed) models from one single simulation run is presented. The algorithm is based on the use of likelihood ratios and related to the Importance Sampling technique for variance reduction. Performing several independent simulation runs of one single model in this manner provides results (expectations of the numbers of molecules of the involved species at any time) for all the considered perturbed models that otherwise had to be simulated separately. The main purposes and the focus of the paper are exhaustive and accurate explanations of the theoretical foundations and the derivation of the simultaneous simulation algorithm itself rather than an excessive presentation of numerical results.

In the remainder of the paper, we proceed in Section 2 with the background and the mathematical specification of stochastic models for biological networks followed by the direct method for stochastic simulation in Section 3, thereby introducing terminology and notation used throughout the paper. This is done a bit more comprehensively than it would be necessary for only presenting the model formulation and the direct simulation algorithm since we also want to shed some light on modeling and simulation of biological networks from a more general Markovian viewpoint. Section 4 provides the necessary theory for the simultaneous simulation algorithm, starting with the foundations of Importance Sampling followed by the derivation of the appropriate framework for biological networks. Then, taking a different perspective on Importance Sampling by reversing the roles of the probability measures involved, it is shown how one can make use of this for simultaneous simulation of multiple models after which the

relevant algorithm is formulated. Runtime comparisons are done in Section 5. Finally, Section 6 concludes the paper and outlines further research directions.

2 Stochastic Modeling of Biological Networks

The basic building blocks of biochemical systems are coupled molecular reactions where the rules at the molecular level are defined in terms of stoichiometric equations. In chemical terminology the fundamental rule of a molecular reaction is given by a stoichiometry

$$s_{m_1} S_{m_1} + \cdots s_{m_r} S_{m_r} \longrightarrow s_{m_{r+1}} S_{m_{r+1}} + + \cdots s_{m_\ell} S_{m_\ell} \tag{1}$$

with $r, \ell \in \mathbb{N},\ r \leq \ell$, where $s_{m_1}, \ldots, s_{m_\ell} \in \mathbb{N}$ are *stoichiometric coefficients*, $S_{m_1}, \ldots, S_{m_r}$ are called *reactants*, $S_{m_{r+1}}, \ldots, S_{m_\ell}$ are called *products* and both reactants and products are *molecular species*. Such a chemical equation expresses that the left hand side of the arrow can be transformed to the right hand side of the arrow. Complex chemical processes are given by sets of such reactions. The stoichiometry thus defines which molecular species may react to result in a certain product and how many molecules are involved in a reaction. The temporal behavior is further specified by assigned *reaction rates*. Several mathematical model approaches reflect different (but of course related) viewpoints and the exact meaning of the reaction rates depends on the chosen model type.

2.1 Stochastic Chemical Kinetics

The application of Markov processes with discrete state space and continuous time to the study of chemical reaction kinetics has a long tradition. It can be traced back to the beginning of the 1940s where the work of Delbrück [11] on autocatalytic reactions was probably the first on the topic. In the 1950s Singer [12] considered chain reactions and some types of coupled reactions, and Bartholomay [13] provided a large body of theory resulting in a series of papers on topics covering sequences of unimolecular and bimolecular reactions, reaction rate constants, and several applications, see [14,15,16] for detailed reviews of the early literature with many more references. These early works already included, though using a different terminology, what is today commonly called the chemical master equation and known to be equivalent to the Kolmogorov differential equations resulting from Markov process theory. Stochastic modeling of chemical reactions by Markov processes is also in accordance with the theory of thermodynamics. A formulation on a physical basis has been provided by Gillespie in [7], [8] and later on rigorously derived in [17]. The basic assumptions are that the system is well stirred and thermally equilibrated, meaning that a well stirred mixture of $d \in \mathbb{N}^+$ molecular species $S_1, \ldots, S_d$ inside some fixed volume interact at constant temperature. In the following we specify the terminology and notation that shall be used throughout the paper.

The system state at any time $t \geq 0$ is a discrete d-dimensional random vector $X(t) = (X_1(t), \ldots, X_d(t))$, where for each species $S_k, k \in \{1, \ldots, d\}$ and $t \geq 0$

a discrete random variable $X_k(t)$ describes the number of molecules of species S_k present at time t. The set $\mathcal{S} \subseteq \mathbb{N}^d$ of all possible system states constitutes the system's state space. The conditional transient (time dependent) probability that the system is in state $x \in \mathcal{S}$ at time t, given that the system starts in an initial state $x_0 \in \mathcal{S}$ at time t_0, is denoted by

$$p^{(t)}(x) := p^{(t)}(x|x_0, t_0) = P\left(X(t) = x \mid X(t_0) = x_0\right). \tag{2}$$

The system changes its state due to chemical reactions between molecules of some species. These reactions can be decomposed into unidirectional reaction channels $R_1, \ldots, R_M$ such that each reaction channel takes the form (1). The reaction rate of each $R_m, m \in \{1, \ldots, M\}$ is given by a well defined function α_m, called the *propensity function* of reaction channel R_m, where $\alpha_m(x)dt$ is the conditional probability that a reaction of type R_m occurs in the infinitesimal time interval $[t, t + dt)$, given that the system is in state x at time t. That is

$$\alpha_m(x)dt = P\left(R_m \text{ occurs in } [t, t + dt) \mid X(t) = x\right). \tag{3}$$

If c_m is the stochastic reaction rate constant[1] assigned to reaction R_m with stoichiometry (1) then the propensity function is simply given by c_m times the number of possible combinations of the required reactants and thus computes as

$$\alpha_m(x) = c_m \cdot \prod_{i=1}^{m_r} \binom{x_{m_j}}{s_{m_j}} \tag{4}$$

where x_{m_j} denotes the number of molecules of species S_{m_j} present in state x, and s_{m_j} is the stoichiometric coefficient according to (1).

Given that the system starts in an initial state $x_0 \in \mathcal{S}$ at time t_0, the temporal evolution of the system is expressed by the *chemical master equation* (CME)

$$\frac{\partial p^{(t)}(x)}{\partial t} = \sum_{m=1}^{M} \left(\alpha_m(x - v_m)p^{(t)}(x - v_m) - \alpha_m(x)p^{(t)}(x)\right) \tag{5}$$

where $v_m = (v_{m1}, \ldots, v_{md})$ is a *state change vector* and v_{mk}, $k \in \{1, \ldots, d\}$ denotes the change of molecules of species S_k due to a reaction of type R_m.

2.2 Relation to Continuous-Time Markov Chains

The propensities are time-independent since the probability that a reaction occurs within a specific time interval only depends on the length of this interval and not on the interval endpoints. Thus, given a current system state, the next state in the system's time evolution only depends on this current system state and neither on the specific time nor on the history of reactions that led to the current

[1] Note that it is easy to convert this stochastic reaction rate constant to/from the rate constant provided by the law of mass actions. Only the volume and the Avogadro number must be appropriately taken into account, see [7,18] for details.

state. Hence, the time evolution of the system is mathematically described by a stochastic process $(X(t))_{t\geq 0}$ with d-dimensional state space $\mathcal{S} \subseteq \mathbb{N}^d$, and due to the just stated independence of time and history this stochastic process is a discrete-state Markov process, in other words it is a Markov jump process, or a continuous-time Markov chain (CTMC). Since terminology and notation in the theory of CTMCs is usually rather different from that used to express the CME, we briefly explain how they correspond to each other. In fact it turns out that the CME is equivalent to the Kolmogorov differential equations which is a much more familiar notion in the general theory of Markov processes.

The multidimensional discrete state space can be mapped to the set $\mathbb{N}$ of nonnegative integers, i.e. each state $x \in \mathcal{S}$ is uniquely assigned to an integer $i \in \{1, \ldots, |\mathcal{S}|\}$. The probability that a transition from state $i \in \mathbb{N}$ to state $j \in \mathbb{N}$ occurs within a time interval of length $h \geq 0$ is denoted by $p_{ij}(h)$, and correspondingly $\mathbf{P}(h) = (p_{ij}(h))_{i,j\in\mathbb{N}}$ is a stochastic matrix, where $\mathbf{P}(0)$ equals the unit matrix $\mathbf{I}$, since no state transitions occur within a time interval of length zero. It is well known (cf. [19,16]) that a CTMC is uniquely defined by an initial probability distribution and a *transition rate matrix*, also referred to as *infinitesimal generator matrix*, $\mathbf{Q} = (q_{ij})_{i,j\in\mathbb{N}}$ consisting of *transition rates* q_{ij} where $\mathbf{Q}$ is the derivative at 0 of the matrix function $h \mapsto \mathbf{P}(h)$. The relation of each $\mathbf{P}(h)$ to $\mathbf{Q}$ and an explanation for the term *infinitesimal generator matrix* is given by $\mathbf{P}(h) = \exp(h\mathbf{Q})$. In that way $\mathbf{Q}$ generates the transition probability matrices by a matrix exponential function which is basically defined as an infinite power series. Hence, all information on transition probabilities is covered by the single matrix $\mathbf{Q}$. In terms of $\mathbf{P}$ and $\mathbf{Q}$ the *Kolmogorov forward differential equations*, the *Kolmogorov backward differential equations*, and the *Kolmogorov global differential equations* can be expressed by (from left to right):

$$\frac{\partial}{\partial t}\mathbf{P}(t) = \mathbf{P}(t)\mathbf{Q}, \quad \frac{\partial}{\partial t}\mathbf{P}(t) = \mathbf{Q}\mathbf{P}(t), \quad \frac{\partial}{\partial t}p^{(t)} = p^{(t)}\mathbf{Q}, \tag{6}$$

where $p^{(t)}$ denotes the vector of the transient state probabilities corresponding to (2). Explicitly writing the Kolmogorov global differential equations in terms of the coefficients and some algebra yields

$$\frac{\partial p_i^{(t)}}{\partial t} = \sum_{j:j\neq i} p_j^{(t)} q_{ji} - \sum_{j:j\neq i} p_i^{(t)} q_{ij} = \sum_{j:j\neq i} \left(p_j^{(t)} q_{ji} - p_i^{(t)} q_{ij} \right). \tag{7}$$

Now, the equivalence of the CME and the Kolmogorov differential equations can be easily seen by interpreting $i \in \mathbb{N}$ as the number assigned to state $x \in \mathcal{S}$, i.e. $p_i^{(t)} = p^{(t)}(x)$, $q_{ij} = \alpha_m(x)$ if j is the number assigned to state $x + v_m$, and $q_{ji} = \alpha_m(x - v_m)$ if j is the number assigned to state $x - v_m$.

3 Direct Stochastic Simulation of Single Models

The essential part of any simulation is to imitate the system under consideration. Consequently, stochastic simulation of biological networks consists of generating

trajectories of a CTMC where the dynamics of CTMCs are imitated. To do so, one has to consider how a CTMC behaves which is in principle known since the days where Markov processes were introduced. When the process is in some state it resides there for an exponentially distributed sojourn time where the mean of this sojourn time is given by the reciprocal of the sum of all outgoing transition rates (independent of the history of the process). When a state transition occurs the probability that the process enters a particular state (the transition probability) is given by the transition rate to that state divided by the aforementioned sum of all outgoing transition rates. Hence, at the transition epochs the CTMC behaves just like a discrete-time Markov chain (DTMC) which enables us to speak of a Markov jump process where a DTMC is embedded at the transition epochs and the times between the transition epochs are exponentially distributed. Though known for a long time before, it seems that Bortz et al [20] were the first who applied this direct method of CTMC trajectory generation to chemical/physical systems. Gillespie [7,8], in addition to the specific formulation of the chemical master equation, also proposed to use stochastic simulation for analyzing coupled chemical reactions where he presented the direct generation of CTMC trajectories in terms of propensity functions as we described in Section 2.1. Therefore, with this terminology it is often called the "Gillespie algorithm" in the biochemical literature, see Algorithm 1 for an according formulation.

Algorithm 1. Direct Method

Init $t := t_0$ and $x := x_0$
repeat
  for all $m = 1, \ldots, M$ **do**
    Compute $\alpha_m(x)$
  end for
  $\alpha_0(x) := \alpha_m(x) + \cdots \alpha_M(x)$
  Generate two random numbers u_1, u_2, uniformly distributed on $(0, 1)$
  Generate time τ to next reaction: $\tau = -\ln(u_1)/\alpha_0(x)$
  Determine reaction type: $m = \min\{k : \alpha_1(x) + \cdots + \alpha_k(x) > u_2\alpha_0(x)\}$
  Update $t := t + \tau;\ x := x + v_m$
  Store/Collect/Handle Data
until terminating condition

Gillespie [7] also discussed an equivalent method that he called the "First Reaction Method" which is due to an equivalent interpretation of the dynamic behavior of CTMCs. Given any state, the tentative times until entering a particular state are all exponentially distributed where the mean is the reciprocal of the according transition rate. Hence, the next state will be entered according to the "fastest" transition rate, i.e. the minimum of the exponentially distributed times. Since the minimum of exponential distributions is again exponentially distributed where the mean is given by the reciprocal of the sum of the parameters (rates) of the involved exponential distributions, the equivalence to the former

interpretation easily follows. Note that this interpretation is often called a *race* in the literature on computer performance evaluation, particularly common in the context of stochastic Petri nets.

Applying this equivalent "race version" of the CTMC dynamics, Gibson and Bruck [21] presented a method called the "Next Reaction Method". Their essential improvement over a naive formulation of the First Reaction Method is simply a more efficient implementation. Roughly speaking, only properties that change are recalculated (others are reused) after a reaction is simulated, and some more advanced data structures are used. For some time this algorithm has been accepted in the according community to be more efficient than the direct method. However, it seems obvious by simply counting the number of necessary operations that this cannot be true. In fact, not surprisingly, Cao, Li and Petzold [22] showed that an optimized implementation of the direct method is indeed more efficient than the Next Reaction Method.

We also like to state that it seems not obvious that it is statistically harmless to reuse calculated random numbers since this may introduce bias to the resulting estimators. To get an intuitive understanding of our doubts, assume that at a certain time in the simulation a random number is generated that implies a very large tentative time of a particular reaction. This can happen even if this reaction is a fast (possibly the fastest) reaction. If random numbers are not reused, it is very likely that in the next steps the tentative times for this reaction are smaller and there is an appropriate probability for this reaction to occur. If random numbers are reused, then it is unlikely that this reaction will ever occur. At the current stage, we are not yet sure if this is a statistically significant effect with regard to the unbiasedness of resulting estimators but it definitely deserves some further investigation in the future. However, the mentioned result of Cao, Li and Petzold implies that the direct method should be preferred anyway. Consequently, we shall choose the direct method as a basis of (statistically) exact stochastic simulation of biological networks.

In any case, it is of course not sufficient to generate only one single trajectory since this only corresponds to one single realization obtained by one specific set of random numbers. Different sets of random numbers imply different trajectories and the resulting estimator, the sample mean, is formally a random variable that has a variance. It is thus necessary to perform sufficiently many independent simulation runs (generate sufficiently many trajectories) such that the variance of the sample mean lies within a reasonable range. In words of Gillespie, it is "necessary to make several simulation runs from time 0 to the chosen time t, all identical with each other except for the initialization of the random number generator" [8]. In fact the reliability of simulation results strongly depends on the estimator's variance.

4 Simultaneous Stochastic Simulation of Multiple Models

In this section, we start with the mathematical foundations of Importance Sampling and the use of likelihood ratios as we shall do in our simultaneous simulation

algorithm. Then we derive an appropriate framework for biological networks and give the necessary reversed view of the roles of the probability measures involved resulting in the simultaneous simulation algorithm.

4.1 Mathematical Foundations of Importance Sampling

Importance Sampling in its basic purpose is a variance reduction technique that makes use of a change of measure and likelihood ratios. The original system is simulated under a different probability measure, and the systematically biased results are weighted by a correcting factor to yield unbiased estimates. The most general description of Importance Sampling is in measure theoretic terms from which all applications to specific model types and domains can be obtained as special cases. Consider two probability measures P and P^* on a measurable space $(\Omega, \mathcal{A})$, where P is absolutely continuous with respect to P^*, that is for all $A \in \mathcal{A}$, $P^*(A) = 0 \;\Rightarrow\; P(A) = 0$. Then, the Radon-Nikodym theorem (cf. e.g. [23,24]) guarantees that the Radon-Nikodym derivative $L = dP/dP^*$ exists and that

$$\forall A \in \mathcal{A} : \; P(A) = \int_A L(\omega) dP^*. \tag{8}$$

In the context of Importance Sampling, the probability measure P^* is called the *Importance Sampling measure*, and the Radon-Nikodym derivative L is usually referred to as the *likelihood ratio*. The basic property exploited by Importance Sampling is that for any random variable Y on $(\Omega, \mathcal{A})$, expectations with respect to P are identical to expectations with respect to P^* when weighting by the likelihood ratio. That is

$$E_P[Y] = \int Y(\omega) dP = \int Y(\omega) L(\omega) dP^* = E_{P^*}[YL] \tag{9}$$

where E_P and E_{P^*} denote expectations with respect to the probability measures P and P^*, respectively.

Hence, when a simulation is performed under P^* the sample mean of YL is an unbiased estimator for $E_P[Y]$. As usual, the variance of the sample mean can be unbiasedly estimated by its sample variance. From that, confidence intervals can be obtained as another important indicator of statistical robustness.

Choosing P^* in place of P is referred to as the *change of measure*, and when the aim is to apply Importance Sampling for variance reduction purposes, it is the essential part and the art of Importance Sampling to perform this change of measure such that estimators with significantly reduced variance are achieved. Many early applications of Importance Sampling, most often concerned with multidimensional Monte Carlo integration, can be found in [25]. The framework for stochastic processes, which is of special interest in our setting of coupled molecular reactions, is given in [26].

It is important to emphasize that Importance Sampling – in the same manner as stochastic simulation in general – is not limited to the estimation of the "raw" expectation of a given objective random variable such as the expected

number of molecules, but it is also applicable to the estimation of expectations of functions of random variables. Equation (9) is valid for any random variable Y. Since real-valued functions of random variables are again random variables, it is possible to estimate the expectation of any real-valued function of a random variable. More concretely, if X is the random variable of interest and one wants to estimate the expectation of $g(X)$ where g is any real-valued function, then one sets $Y = g(X)$ and applies equation (9). Explicitly rewritten, in this context equation (9) becomes

$$E_P[g(X)] = \int g(X(\omega))dP = \int g(X(\omega))L(\omega)dP^* = E_{P^*}[g(X)L]. \tag{10}$$

Although, as explained, already covered by (9), the generality and flexibility of Importance Sampling may be more clearly expressed by (10). Particularly important cases are arbitrary moments of X that are simply powers of X, mixtures of moments with the variance as one special case, and probabilities of arbitrary events that can be estimated via expectations of indicator functions. In the context of Markov processes, the random variable X describes a path of the process, and $Y = g(X)$ may be any function on a path. For instance, in the simple setting of estimating the expected number of molecules of species S_k at time t we have $Y = g(X) = X_k(t)$, but many more properties can be considered. For Markov processes it may be of interest how often certain specific transitions (reaction types in the context Section 2.1) occur, how often certain sets of states are reached, or at what time certain (possibly absorbing) states are reached for the first time. For stochastic chemical kinetics the latter corresponds to the question how long it takes until molecules of certain species are exhausted.

4.2 Importance Sampling for Biological Networks

For CTMCs, the probability measures P and P^* are path distributions and absolute continuity corresponds to the condition that all paths that are possible under P, that is in the original model, must remain possible under P^*. In continuous time this can be obviously achieved by the condition that for all positive transition rates in the original model the corresponding transition rates under Importance Sampling are positive. Since we deal with CTMCs given in terms of biochemical notation as described in Section 2.1, we need an appropriate framework for the application of Importance Sampling to that type of models. Applying Importance Sampling to coupled molecular reactions first of all requires the distribution or density, respectively, of reaction paths that we will derive in the following.

The discrete state of the system changes due to molecular reactions. Let $t_1 < t_2 < \ldots$ denote the successive time instants at which reactions occur and R_{m_i} the reaction type that occurs at time t_i, where $m_i \in \{1, \ldots, M\}$. Define $\tau_i := t_{i+1} - t_i$ the time between the i-th and the $(i+1)$-th reaction. Hence, state $x(t_i)$ is reached

due to the i-th reaction R_{m_i} at time t_i and remains unchanged for a sojourn time of τ_i after which the $(i+1)$-th reaction $R_{m_{i+1}}$ occurs at time t_{i+1} and changes the state to $x(t_{i+1})$. Thus, the time evolution of the system is completely described by the sequence of states and corresponding sojourn times, and in compact form $(x(t_0), \tau_0), (x(t_1), \tau_1), (x(t_2), \tau_2), \ldots$ describes a trajectory. For a trajectory up to the R-th reaction, considering the Markovian property which in turn implies exponentially distributed sojourn times, the reaction path density is given by

$$dP((x(t_0), \tau_0), \ldots, (x(t_R), \tau_R)) =$$

$$p^{(t_0)}(x_0) \cdot \prod_{i=1}^{R} \alpha_{m_{i-1}}\left(x(t_{i-1})\right) \exp\left(\alpha_0(x(t_{i-1}))\tau_{i-1}\right) \qquad (11)$$

where $\alpha_0(x(t_{i-1})) := \alpha_1(x(t_{i-1})) + \cdots \alpha_M(x(t_{i-1}))$, similarly as explained before and used in the direct generation of CTMC trajectories. Note that for a given time horizon over which the system is observed (and should be simulated) the number R of reactions is not known in advance and in particular not deterministic. Formally, R is a random stopping time[2] which is in accordance with the requirement of dP being a density of a probability measure P defined on the path space of the Markov process.

Now, in order to perform an Importance Sampling simulation, we need to change the underlying probability measure, which is in the case of coupled molecular reactions determined by the propensity functions. Since the only requirement for the application of Importance Sampling is absolute continuity of the probability measures involved, there is a great freedom in how to change the measure. It is only necessary that all reaction paths that are possible (have positive probability) under the original measure remain possible. That means each probability measure on the path space that meets the aforementioned condition can be considered, even non-Markovian models are allowed as long as they assign positive probabilities to all possible reaction paths.

Nevertheless, we should avoid a large increase in trajectory generation efforts compared to the original measure. Thus, obviously the most natural (and valid) change of measure is to remain in the Markovian world and the easiest way is to simply change the original propensity functions to "Importance Sampling propensity functions" α_m^* such that for all $m \in \{1, \ldots, M\}$ we have $\alpha_m^*(x) = 0 \;\Rightarrow\; \alpha_m(x) = 0,\ x \in \mathcal{S}$, or equivalently, starting with the original propensity functions, $\alpha_m(x) > 0 \;\Rightarrow\; \alpha_m^*(x) > 0,\ x \in \mathcal{S}$. Importance Sampling then generates trajectories according to the changed propensity functions and multiplies the results with the likelihood ratio to get unbiased estimates for the original system. The trajectory generation is thereby performed as before, e.g. by applying the direct method, where now the changed propensity functions are used, yielding a sequence of states with associated sojourn times and reaction

[2] The term *random stopping time* does not only refer to random variables that have a "real-life interpretation" as times but its definition is a general one in the context of random variables, see e.g. [19,23,24] for formal definitions.

path density as in (11). Thus, denoting by $p^{*(t_0)}$ the initial distribution for the states, the likelihood ratio becomes

$$L(\omega) = \frac{p^{(t_0)}(x_0) \cdot \prod_{i=1}^{R} \alpha_{m_{i-1}}(x(t_{i-1})) \exp(\alpha_0(x(t_{i-1}))\tau_{i-1})}{p^{*(t_0)}(x_0) \cdot \prod_{i=1}^{R} \alpha^*_{m_{i-1}}(x(t_{i-1})) \exp(\alpha^*_0(x(t_{i-1}))\tau_{i-1})}. \tag{12}$$

Rewriting this likelihood ratio yields

$$L(\omega) = \frac{p^{(t_0)}(x_0)}{p^{*(t_0)}(x_0)} \cdot \prod_{i=1}^{R} \frac{\alpha_{m_{i-1}}(x(t_{i-1})) \exp(\alpha_0(x(t_{i-1}))\tau_{i-1})}{\alpha^*_{m_{i-1}}(x(t_{i-1})) \exp(\alpha^*_0(x(t_{i-1}))\tau_{i-1})} \tag{13}$$

which shows that the likelihood ratio can be efficiently computed during trajectory generation without much extra computational effort by successively updating its value after each simulated reaction according to the running product. In particular, the unbiased number of molecules can be obtained at any time. In the case of a given initial state (initial numbers of molecules of each species) all the probability mass is centered on this state, thus the initial distribution assigns the probability of one to the initial state and zero to all other states. If we keep the same initial state as in the original model under Importance Sampling, the initial distribution remains unchanged, too, which means that the initial distributions in the original model and under Importance Sampling both assign the probability of one to the initial state and zero to all other states. Thus, the ratio of the probabilities of the initial state is one, and the likelihood ratio becomes

$$L(\omega) = \prod_{i=1}^{R} \frac{\alpha_{m_{i-1}}(x(t_{i-1})) \exp(\alpha_0(x(t_{i-1}))\tau_{i-1})}{\alpha^*_{m_{i-1}}(x(t_{i-1})) \exp(\alpha^*_0(x(t_{i-1}))\tau_{i-1})} \tag{14}$$

which again can be efficiently updated after each simulated reaction.

Although naturally arising, the change of measure as described above may be too restrictive in a variance reduction setting. In cases where more flexibility is needed, it is possible to use a different change of measure in each simulation step or propensity functions that depend on the number of already occurred reactions (corresponding to a nonhomogeneous model) or the history of the just executed simulation steps. Formally, define functions $\beta^{(r)}_m(x(t_0), \ldots, x(t_r))$, where for all $m \in \{1, \ldots, M\} : \alpha_m(x(t_r)) > 0 \Rightarrow \beta^{(r)}_m(x(t_0), \ldots, x(t_r)) > 0$. Then the reaction path density under Importance Sampling is

$$\begin{aligned} &dP((x(t_0), \tau_0), \ldots, (x(t_R), \tau_R)) = \\ &p^{(t_0)}(x_0) \cdot \prod_{i=1}^{R} \beta^{(i-1)}_{m_{i-1}}(x(t_0), \ldots, x(t_{i-1})) \exp(\beta_0(x(t_0), \ldots, x(t_{i-1}))\tau_{i-1}) \end{aligned} \tag{15}$$

and the corresponding likelihood ratio (leaving the initial distribution unchanged) becomes

$$L(\omega) = \prod_{i=1}^{R} \frac{\alpha_{m_{i-1}}(x(t_{i-1})) \exp(\alpha_0(x(t_{i-1}))\tau_{i-1})}{\beta^{(i-1)}_{m_{i-1}}(x(t_0), \ldots, x(t_{i-1})) \exp(\beta_0(x(t_0), \ldots, x(t_{i-1}))\tau_{i-1})}. \tag{16}$$

However, the latter form of the change of measure is more involved than the straightforward one, and our purpose in this paper is not to reduce the variance of the estimator but to provide an efficient method for comparisons of multiple parameter settings that need not be chosen but are given through the objective of the study. Therefore, we shall rely on the change of measure that preserves the Markovian structure of the set of coupled reactions.

4.3 Reversing Roles in Importance Sampling

Originally, Importance Sampling was not intended for comparing multiple models but for reducing the variance of the simulation estimator for a single model. However, if we reverse the roles of the original distribution and the Importance Sampling distribution, we can elegantly obtain estimates for (in principle arbitrarily many) models simultaneously from one single simulation experiment (as usual consisting of independent simulation runs).

Consider the nominal model, that is the model assumed to be the appropriate unperturbed one, as the Importance Sampling model. That means, according to the usual procedure of Importance Sampling, the simulation will be performed only with the dynamics (according to the reaction rates and propensity functions) of that model. Then any perturbed model can take the role of the original model in the equations derived in Section 4.2 and applying Importance Sampling means to simulate the nominal model and weighting the outcomes by the likelihood ratio to obtain unbiased estimates for the perturbed model. It is important that this is valid for any perturbed model where the stoichiometry is the same as for the unperturbed (original) model. Hence, arbitrary changes/perturbations of the rate constants and thus the propensity functions can be considered.

For simultaneously simulating more than one perturbed model we keep track of multiple likelihood ratios resulting from the perturbed models and update them all after each simulated reaction for the nominal model, similarly as suggested by (13) and (14), respectively. In this paper we consider perturbations in the reaction rates and keep the initial numbers of molecules unperturbed which seems very reasonable since the sources of modeling errors are usually the reaction rates that rely on, e.g., observations from laboratory experiments or similar experiences, whereas initial numbers of molecules are controlled parts of experimental settings. Thus, the appropriate form of the likelihood ratio for us here is (14) and we are now ready to formulate an algorithm based on the theoretical derivations of the previous sections.

Note that such an algorithm involves trajectory generation of the nominal model and thus naturally relies on an according algorithm for this purpose and extends this algorithm by incorporating the likelihood ratio computations. Hence, the main effort in developing a method for simultaneous stochastic simulation of many models and showing its correctness has been already done with our previous theoretical derivations. Now, in principle, any suitable algorithm for trajectory generation can be used, e.g. the direct method, the First Reaction Method, the Next Reaction Method and even any version of tau leaping or other approximate algorithms, and equipped with the feature of Importance Sampling.

As mentioned in Section 3, we choose the direct method for trajectory generation. We assume that N perturbed models should be simulated simultaneously. The propensity functions of the nominal model are denoted by α_m^* and the propensity functions of the perturbed models by $\alpha_m^{(1)}, \ldots, \alpha^{(n)},\ m = 1, \ldots, M$ where, as before, M is the number of reaction channels. Since all perturbed models follow the same stoichiometry as the nominal model, they all have the same state change vectors as the nominal model. With these notation, Algorithm 2 performs the simultaneous simulations.

Algorithm 2. Simultaneous Stochastic Simulation Algorithm

Init $t := t_0, x := x_0, L_1 = \cdots = L_N = 1$
repeat
  for all $m = 1, \ldots, M$ **do**
    Compute $\alpha_m^*(x)$
  end for
  $\alpha_0^*(x) := \alpha_m^*(x) + \cdots \alpha_M^*(x)$
  for all $i = 1, \ldots, N$ **do**
    for all $m = 1, \ldots, M$ **do**
      Compute $\alpha_m^{(i)}(x)$
    end for
    $\alpha_0^{(i)}(x) := \alpha_1^{(i)}(x) + \cdots + \alpha_M^{(i)}(x)$
  end for
  Generate two random numbers u_1, u_2, uniformly distributed on $(0, 1)$
  Generate time τ to next reaction: $\tau = -\ln(u_1)/\alpha_0^*(x)$
  Determine reaction type: $m = \min\{k : \alpha_1^*(x) + \cdots + \alpha_k^*(x) > u_2\alpha_0^*(x)\}$
  for all $i = 1, \ldots, N$ **do**
    Update $L_i = L_i \cdot \alpha_m^{(i)}(x)/\alpha_m^*(x) \cdot \exp((\alpha_0^*(x) - \alpha_0^{(i)}(x)) \cdot \tau)$
  end for
  Update $t := t + \tau;\ x := x + v_m$
  Store/Collect/Handle Data
until terminating condition

As can be easily seen, compared to Algorithm 1 the extra effort is in the necessary computation of all propensity functions of all perturbed models and the update of the likelihood ratios that have to be done after each simulated reaction of the nominal model (inside the repeat loop). Neither additional random numbers must be generated nor additional reactions must be simulated.

5 Runtime Comparison

In order to demonstrate the savings in runtime yielded by simultaneous simulation, we consider the estimation of expected numbers of molecules. Note that, as explained in Section 4.1, this is only one of the many possible applications. However, the extra effort is independent of the objective and it is thus the same for other objectives like, e.g., variances, higher moments, or absorption times.

Hence, for the purpose of demonstrating the efficiency gain it is sufficient to consider the estimation of expected numbers of molecules.

Given that we rely on the direct method for generating trajectories it does not make sense to present comparisons of the sample means of trajectories generated by separate simulation experiments with the direct method and simultaneous simulations weighted by likelihood ratios. Not surprisingly, since - as we have shown - the appropriately weighted results from simultaneous simulations and the results from separate simulations are both unbiased estimators of the expected numbers of molecules at any time, they should not show statistically significant differences if a sufficient number of simulation runs is performed. This theoretical consideration was indeed also observed in practice when verifying the implementation with a couple of different models.

Of course, it is of interest how the extra effort affects the runtime. If this extra effort for each model was in the order of the effort of simulating a single model separately, simultaneous simulations would provide no gain. Hence, what must be demonstrated is that the overhead due to the likelihood ratio updates is significantly smaller than the effort for a single simulation. Note that our purpose here is not a comprehensive study of a specific large model but the demonstration of the runtime gain provided by the simultaneous stochastic simulation algorithm. Of course, the algorithm is valid for any stochastic model of biological networks in the framework of Section 2.1, of any size and complexity, and there are no additional requirements or restrictions on the model structure.

As a representative example that we systematically studied we present runtimes obtained for the decaying dimerization reaction set

$$S_1 \xrightarrow{c_1} \emptyset, \quad S_1 + S_1 \underset{c_3}{\overset{c_2}{\rightleftharpoons}} S_2, \quad S_2 \xrightarrow{c_4} S_3$$

that was also chosen for instance in [9,27] to study the performance of different tau-leaping methods. The parameter choices were taken in accordance with the mentioned references as $X_1(0) = 400,\ X_2(0) = 798,\ X_3(0) = 0$ for the initial numbers of molecules and $c_1 = 1, c_2 = 10, c_3 = 1000, c_4 = 0.1$ for the rate constants. Thus the propensity functions are given by $\alpha_1(x) = x_1,\ \alpha_2(x) = 5x_1(x_1 - 1),\ \alpha_3(x) = 1000x_2,\ \alpha_4(x) = 0.1x_2$. The model was simulated up to time horizon $t = 0.2$ and up to time horizon $t = 0.5$ For both time horizons we performed two series of simulations of perturbed models In the first series exactly one of the reaction rates was perturbed by $\pm 3\%$. Hence, altogether eight parameter settings were considered. In the second series, all reaction rates were considered to be potentially perturbed by $\pm 3\%$ at the same time. Hence, including the nominal one, altogether $3^4 = 81$ parameter settings were considered, i.e. 80 differently perturbed parameter settings.

Table 1 contains the runtimes of matching simulations with the direct method and the simultaneous method. With the direct method 8 and 80, respectively, separate simulation experiments, each consisting of 10^4 simulation runs up to the given time horizons, were done according to Algorithm 1. With the simultaneous method the 8 and the 80, resp., perturbed parameter settings were simulated with 10^4 simulation runs according to Algorithm 2. Not surprisingly, the runtime

Table 1. Runtime Comparisons for the Decaying Dimerization Reaction with Multiple Perturbations. Runtimes are given in seconds.

	$t = 0.2$		$t = 0.5$	
# Perturbed Models	Direct	Simult	Direct	Simult
8	28928	8178	72504	18528
80	290155	51449	725871	99303

of the direct method for N perturbed parameter settings is roughly N times the runtime for one single setting. With the simultaneous method the runtime grows significantly less with increasing number of parameter settings. This shows that the extra effort according to the likelihood ratio computation is far less than the effort for a separate simulation. More formally and generally, let r denote the runtime for one separate simulation experiment (with sufficiently many independent runs) for the nominal parameter settings[3] with the direct method. Then the runtime for simulating N parameter settings separately is $R_{\text{sep}} \approx Nr$. For the simultaneous method the runtime is $R_{\text{simult}} = r + Nv$ where v denotes the runtime due to the overhead (extra effort) for one additional parameter setting. Obviously, when only one parameter setting is considered $R_{\text{sep}} = r < r + v = R_{\text{simult}}$ which simply means that in this case of course no extra effort should be introduced. In general,

$$R_{\text{simult}} < R_{\text{sep}} \;\Leftrightarrow\; v < r - \frac{r}{N}. \tag{17}$$

Hence, we have a condition under which the simultaneous method is faster. This condition is very likely to be met for all models, of any size and complexity. As a final remark we state that we performed (less systematical) runtime comparisons for several different reference reaction sets with a moderate number (up to ten) of reaction channels.

6 Conclusion

We have presented an algorithm for simultaneous simulation of multiple parameter settings within one single simulation experiment. The algorithm is based on likelihood ratios that appropriately weight the simulation results to obtain unbiased estimates. The algorithm is particularly useful for comparisons of a large number of parameter settings and it yields a large amount of runtime gain compared to multiple separate simulations. Thereby, no approximation and thus no loss in statistical validity and accuracy is introduced. Further research includes analytical investigations of the algorithm performance as well as the incorporation of other methods for trajectory generation such as, e.g., tau leaping.

[3] For small parameter perturbations this runtime is roughly the same for all perturbed parameter settings.

References

1. Arkin, A., Ross, J., McAdams, H.H.: Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected escherichia coli cells. Genetics 149, 1633–1648 (1998)
2. Blake, W.J., Kaern, M., Cantor, C.R., Collins, J.J.: Noise in eukaryotic gene expression. Nature 422, 633–637 (2003)
3. Fedoroff, N., Fontana, W.: Small numbers of big molecules. Science 297, 1129–1131 (2002)
4. McAdams, H.H., Arkin, A.: Stochastic mechanisms in gene expression. Proceedings of the National Academy of Science USA 94, 814–819 (1997)
5. McAdams, H.H., Arkin, A.: It's a noisy business! Trends in Genetics 15(2), 65–69 (1999)
6. Turner, T.E., Schnell, S., Burrage, K.: Stochastic approaches for modelling in vivo reactions. Computational Biology and Chemistry 28, 165–178 (2004)
7. Gillespie, D.T.: A general method for numerically simulating the time evolution of coupled chemical reactions. Journal of Computational Physics 22, 403–434 (1976)
8. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 71(25), 2340–2361 (1977)
9. Gillespie, D.: Approximate accelerated stochastic simulation of chemically reacting systems. Journal of Chemical Physics 115, 1716–1732 (2001)
10. Gunawan, R., Cao, Y., Petzold, L.R., Doyle III, F.J.: Sensitivity analysis of discrete stochastic systems. Biophysical Journal 88, 2530–2540 (2005)
11. Delbrück, M.: Statistical fluctuations in autocatalytic reactions. Journal of Chemical Physics 8, 120–124 (1940)
12. Singer, K.: Application of the theory of stochastic processes to the study of irreproducible chemical reactions and nucleation processes. Journal of the Royal Statistical Society 15, 92–106 (1953)
13. Bartholomay, A.F.: A Stochastic Approach to Chemical Reaction Kinetics. Thesis, Harvard University (1957)
14. Bharucha-Reid, A.T.: Elements of the Theory of Markov Processes and Their Applications. McGraw-Hill, New York (1960)
15. McQuarrie, D.A.: Stochastic approach to chemical kinetics. Journal of Applied Probability 4, 413–478 (1967)
16. van Kampen, N.: Stochastic Processes in Physics and Chemistry. Elsevier, Amsterdam (1992)
17. Gillespie, D.T.: A rigorous derivation of the chemical master equation. Physica A 188, 404–425 (1992)
18. Wolkenhauer, O., Ullah, M., Kolch, W., Cho, K.H.: Modelling and simulation of intracellular dynamics: Choosing an appropriate framework. IEEE Transactions On NanoBioScience 3(3), 200–207 (2004)
19. Bremaud, P.: Markov Chains. Springer, Heidelberg (1999)
20. Bortz, A.B., Kalos, M.H., Lebowitz, J.L.: A new algorithm for Monte Carlo simulation of Ising spin systems. Journal of Computational Physics 17, 10–18 (1975)
21. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. Journal of Physical Chemistry A 104, 1876–1889 (2000)
22. Cao, Y., Li, H., Petzold, L.R.: Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. Journal of Chemical Physics 121(9), 4059–4067 (2004)

23. Feller, W.: An Introduction to Probability Theory and its Applications, vol. 2. Wiley, Chichester (1971)
24. Shiryaev, A.N.: Probability, 2nd edn. Springer, Heidelberg (1995)
25. Hammersley, J.M., Handscomb, D.C.: Monte Carlo Methods. Methuen (1964)
26. Glynn, P.W., Iglehart, D.L.: Importance sampling for stochastic simulations. Management Science 35(11), 1367–1392 (1989)
27. Cao, Y., Gillespie, D., Petzold, L.: Avoiding negative populations in explicit Poisson tau-leaping. Journal of Chemical Physics 123, 54104 (2005)

Modelling Yeast Pre-rRNA Processing

Federica Ciocchetta[1], Jane Hillston[1], Martin Kos[2], and David Tollervey[2]

[1] Laboratory for Foundations of Computer Science, The University of Edinburgh, Edinburgh EH9 3JZ, Scotland
[2] The Wellcome Trust Center for Cell Biology, The University of Edinburgh, Edinburgh EH9 3JR, Scotland

Abstract. In this paper we present a quantified model concerning the synthesis of pre-rRNAs. The chemical kinetics simulation software Dizzy has been chosen as both the modelling and simulation framework of our study. We discuss the validation of the model against the available experimental data and we show some preliminary results obtained from the study of our model. All the analyses are based on stochastic simulation.

1 Introduction

Eukaryotic cells contain a huge variety of RNA species, almost all of which are synthesised by post-transcriptional processing [1,2]. The Tollervey Lab at the University of Edinburgh is investigating the mechanisms and regulation of RNA processing and turnover, using yeast as a model organism. In this paper we present a quantified model of one such synthesis pathway in yeast, pre-rRNA processing. We are particularly interested in the relative frequency of two alternative forms of the pathway. In the first pathway a complete pre-rRNA is formed and processed in a series of cleavage and modification steps. In the second variant, known as *co-transcriptional cleavage (CoTC)*, part-formed pre-rRNA is cleaved from the nascent transcript and begins cytosolic modification, whilst the remainder continues transcription. The dynamics of the processing is studied using stochastic simulation in the Dizzy tool [7,8].

In this paper we present the construction of a computational model which encompasses both forms of the pre-rRNA processing and explain the subtleties required in order to validate the model against experimental data. The intermediate products of the pre-rRNA processing are not readily measurable unless labelled by radioactive uracil. Thus the experimental data is presented in terms of labelling intensity, rather than directly recorded amounts of the precursors. As a consequence the labelling mechanism must also be incorporated into the computational model.

This lack of accessibility of the biological entities are one of the main motivations for using computational models to support the wet lab experiments for this system. The Dizzy software package [7,8] has been used as the modelling and analysis framework of our work. A stochastic simulation, based on Gillespie's algorithm, was chosen as it allowed us fine-grained control over elements of primary importance such as the relative frequency of co-transcriptional cleavage.

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 32–47, 2007.

We also present the validation of the model against the available experimental data and preliminary experiments investigating the sensitivity of the accumulation of the various intermediate products to different proportions of DNAs exhibiting co-transcriptional cleavage.

There is a substantial literature modelling yeast and/or RNA. In particular we mention [3,4,5]. In [3] a kinetic-dynamic model was proposed to simulate RNA processing by determining the essential reaction rates, including the rates of transcription, pre-mRNA turnover, pre-mRNA splicing, and mRNA decay. A simulator based on the family competition evolutionary algorithm was applied on several artificial datasets and on a simplified yeast expression dataset. The authors of [4] presented and analysed a model of protein translation at the scale of an individual messenger RNA (mRNA) transcript. In [5] the authors proposed the development of mathematical models that quantitatively describe the complex process of transcription, RNA processing, transport, translation and mRNA turnover.

The study of the synthesis pathway presented in this paper is novel. Also the use of Dizzy for modelling and analysis in this field is new.

Structure of this paper. In Section 2 we present the biological model on which our study is based. We follow this in Section 3 with a detailed presentation of the computational model, developed for the stochastic simulator, Dizzy. In Section 4 we present a comparison of the output of our model and the available experimental data. Finally, we present conclusions in Section 5.

2 Biological Model

RNA plays a fundamental role in the translation of genes into proteins. In addition to the messenger RNA (mRNA) which is transcribed from the gene and serves as a template for the protein formation, the translation itself takes place on ribosomes which may be thought of as molecular machines comprised of several molecules of ribosomal RNA (rRNA). Whilst mRNA is specific to the particular protein being translated, ribosomes and the associated rRNA are more general, translating various mRNA. For this reason vast numbers of ribosomes, and therefore also rRNA molecules, are needed.

The rRNA molecules are transcribed from DNA in the same way as mRNA. RNA polymerase moves along the DNA which can be regarded as a template, from the 5' end to the 3' end (left to right in our diagrams), joining together the nucleotides in the order specified by the template. Thus as the polymerase progresses along the DNA a growing chain of nucleotides is formed. Because rRNA is required in large amounts within the cell there are multiple copies of the genes, arranged in tandem along the genome. So at any particular time for each gene there will be numerous partially formed rRNA molecule, at different lengths/stages of synthesis and this will be repeated on the many copies of the gene within the cell. This situation is illustrated in Fig. 1. These partially formed rRNA molecules are termed *nascent transcripts* (NT).

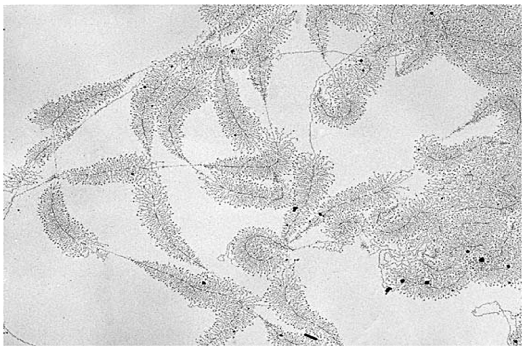


Fig. 1. Visualisation of rRNA transcription (from [6]). Each feather-like structure consists of a ribosomal DNA (the central spine) and a number of actively transcribed rRNA units (polymerase chains) of increasing length.

At the end of transcription the molecule formed is termed pre-rRNA, or a precursor, because it will be subject to further modification before it is ready for incorporation into the ribosome. The processing which these pre-rRNA molecules undergo is an area of active research. For yeast cells, quite a lot of detail is known about the processing and many intermediate stages have been detected experimentally. The intermediate elements are measured in terms of Svedberg units or S values, which are a measure of the rate at which they sediment out in an ultracentrifuge.

In this paper we consider the processing pathway from the fully formed precursor 35S shown in Fig. 2. In addition to the reactions shown, 35S may split into 23S and 27SB (*react4*). We consider the possibility that the polymerase chain does not remain intact for the whole transcription, but instead completed elements of 20S become detached, whilst transcription is still in progress. This is termed *co-transcriptional cleavage* (CoTC) and is illustrated in Fig. 3. Whilst this alternative behaviour is known to occur there is no consensus about how common it is with respect to transcription without intermediate cleavage. Moreover, it is also unknown at what stage of transcription the cleavage occurs. For example, it could occur as soon as the polymerase chain contains a complete 20S element, but also at any point later in the chain development. Answering these questions has been a major motivation for the development of the computational model which we present in this paper.

3 The Dizzy model

In this section we give an account of the model which we have developed. We used the Dizzy software package [7,8], and so the section starts with a brief introduction to that package. The remainder of the section is devoted to a description of the model. In particular we focus on the two major challenges that we faced in developing the model:

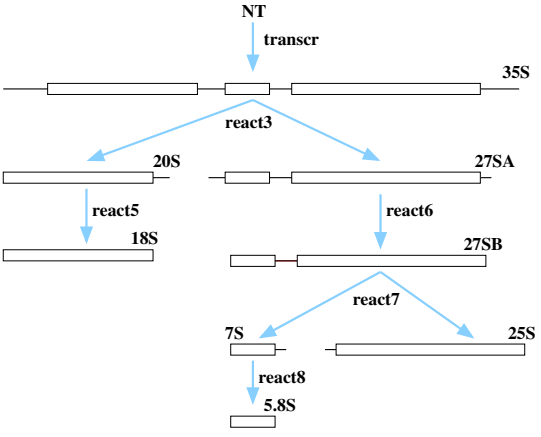


Fig. 2. The ribosome synthesis pathway, without co-transcriptional cleavage

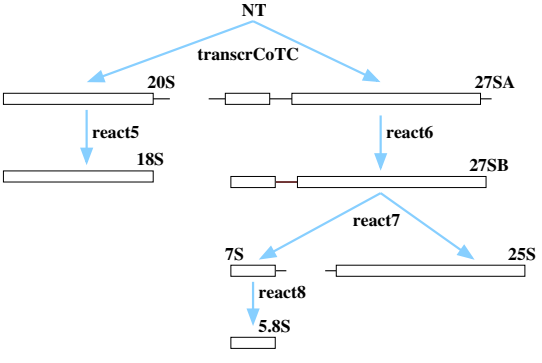


Fig. 3. The ribosome synthesis pathway, following co-transcriptional cleavage

- adequately capturing the choice between the usual *transcription* and *CoTC* to allow suitable experimentation;
- representing the *labelling* process, especially with respect to the initial period, as this is intrinsic in the experimental data.

We consider only one cell, as each cell is supposed to act independently for this process. Within each cell there are approximately 100 copies of the rRNA gene [15], which we will term *rDNA sequences*, responsible of the pre-rRNA transcription.

3.1 Dizzy

Dizzy [7,8] is a chemical kinetics simulation software package written in Java. It provides a model definition environment and an implementation of both

stochastic and deterministic simulation algorithms. The choice of Dizzy was motivated by the fact that our analysis is founded on stochastic simulation and Dizzy offers the implementation of well-known and widely-used simulation algorithms. Furthermore, its simple input language, defined for modelling biological systems, allowed us to represent our model in an effective and straightforward way, at the level of details we are interested in. Indeed, as we describe below, our study is based on DNA intervals of around *400* bases (the length of DNA transcripted in the time of the pulse-labelling experiments) instead of the single basis and this can be easily modelled in Dizzy.

Dizzy uses the *Chemical Model Definition Language (CMDL)* as input language. A CMDL model definition consists of a series of statements that define the model elements, such as *species* and *reactions*, but also *parameters* and *compartments*. In the former part of a CMDL model there are definitions of the species and possibly of compartments and parameters. Each is described by a *symbol name* associated with a *value* (the initial amount for species). This is followed by *reaction statements*. A reaction statement defines a one-way chemical reaction involving zero or more reactants, and zero or more products. Reaction statements have three elements: reaction name, the list of reactants and of products, and the reaction rate.

Among the different simulation implementations offered by Dizzy [10,11,12,9,16], in this work we chose Gillespie's Direct Method for the analysis. This uses the Monte Carlo technique to generate an approximate solution of the master equation for chemical kinetics. Broadly speaking, the algorithm tracks the evolution of the system starting from an initial state by computing the time and the kind of the next reaction by means of two probability density functions. At each step the global state is updated and the procedure repeats until the simulation time is reached or no further reactions can be fired.

3.2 Usual transcription and CoTC

A main process in the model is the transcription of pre-rRNAs from the rDNA sequence. We abstract away from the simple elementary steps that compose transcription and so consider it as a unique biological process. We model both the alternative pathways reported in Figs. 2 and 3.

The mechanism of selection between the usual transcription and CoTC is still not completely known, nor the frequency with which the two alternatives occur. In order to model the choice between the two different kinds of transcription the frequencies $p1$ and $p2$ are introduced, representing the probabilities to have the usual transcription and CoTC, respectively ($p2 = 1 - p1$). Furthermore we divide the rDNA sequences in the cell into two groups. The elements of the former group, indicated by the name *DNA*, are involved in the usual transcription and the elements of the latter group, indicated by $DNACoTC$, in CoTC. Since in one cell there are about 100 rDNA sequences involved in the rRNA synthesis [15], we have globally $100 * p1$ DNAs and $100 * p2$ DNACoTC. We must also account for the partially formed transcripts. Thus, in addition, we have to consider $100 * p2$ DNAp27SA2, which represent the partial transcription of 27SA2 obtained

from cleavages that have happened before the initial time. The values of the two frequencies are hypothesised to be 0.7 and 0.3 from the current biological knowledge of the process, but these values must be checked. In the analysis of the system, the parameters $p1$ and $p2$ are varied to study the influence that the kind of transcription selected has on the evolution of (the concentration of) some species.

The exact point at which the CoTC starts is unknown. In the present model it is supposed that CoTC happens soon after the transcription of the 20S (end of region 11). However it is possible to consider different possibilities and see how this influences the production of the pre-rRNAs.

3.3 Labelling

Broadly speaking, the labelling describes the process during which radioactive uracil, a base nucleotide, is introduced into the cell and is incorporated in the pre-rRNA during transcription. The resulting elements are labelled and can be detected in the experiments.

The following hypotheses concerning the labelling are made:

1. the radioactive uracil is introduced into the cell at time $t_0 = 0$;
2. the labelling equilibration is very fast (a few seconds) and we do not need to consider it;
3. the radioactive uracil is in large quantity and does not lose radioactivity (at least for the time considered). So we can suppose a constant supply of it: all additions to polymerase chains after the introduction of the label will incorporate it.

In order to model the labelling we need to distinguish between the initial period, during which the nascent transcripts exhibit different levels of labelling according to their stage of transcription when the labelling was introduced, and the situation after some time, in which only fully-labelled elements are obtained.

During the initial period, we distinguish different states of the nascent transcripts at time t_0. Indeed, different initial transcription situations lead to different levels of labelling in the resulting pre-rRNAs. For instance, if at time t_0 a transcription has almost completed the resulting transcript will be only minimally labelled. On the other hand, if the transcription started just before t_0 a fully labelled element will be obtained.

To model the different steps in the transcription and the resulting different levels in the labelling, the whole DNA sequence of interest has to be discretized into regions. It is split into *400*-base regions, corresponding to the bases transcribed in the period of 10 s. This corresponds to the time period of observations in the pulse-labelling experiments. As the total sequence has a length of about 6660 bases, *17* regions are obtained. We use the index i to indicate the different regions, starting from 1 (rightmost region) to 17 (leftmost region). They correspond to the different levels of labelling (from minimum to maximum) of the resulting pre-rRNA.

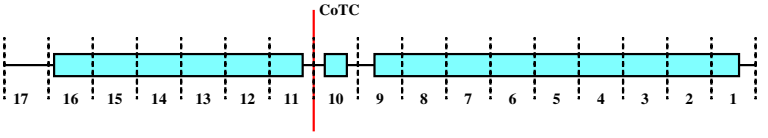


Fig. 4. Discretization of the rDNA sequence in 17 regions of 400b

Figure 4 reports a schema of the discretization. It is also shown the position of CoTC, that in the present model is fixed at the end of region 11 (end of 20S).

The various pre-rRNA and rRNA involve different numbers of regions. For instance 20S and 18S involve the regions from 16 to 11, 5.8 and 7S the region 10, while 25S involves the regions from 9 to 1. The elements 27SA2 and 27SB cover the regions from 10 to 1.

In the case of the intermediate elements, the level of labelling depends on the labelling of the initial pre-rRNA from which it is derived and on which regions are present in the element. For instance in the case of 35S we can have 17 states of labelling. The pre-rRNA 20S has 6 levels of labelling plus the unlabelled state, obtained when the 35S from which 20S is derived is labelled up to level 10.

We use the suffix l_i to refer to the initial transcription situation and to indicate the level of labelling of the different elements. The maximum level refers to the situation beyond the initial labelling period.

3.4 Description of the model

In this section we give a brief description of the Dizzy model.

Frequencies p1 and p2. The variable $p1$ and $p2$ are introduced to describe the frequencies of the two kinds of transcription. We have the statements:

$p1 = 0.7$;
$p2 = 0.3$;

Species and initial amounts. Each species used in the reaction must be defined in the model. We have two main groups of species: the ones involved in the transcription and the ones representing the intermediate and final elements of the pathways. For the former kind, we have the following species:

- DNA_{l_i} for $i = 1, 2, ..., 17$;
- $DNACoTC_{l_i}$ for $i = 11, 12, ...17$;
- $DNAp27SA_{l_i}$ for $i = 1, 2, .., 10$.

The former species represent the case of nascent transcripts whose transcription is initially in the region i and the transcription is the usual one. $DNACoTC_{l_i}$ has a similar meaning, but in the case of CoTC. This only considers 7 regions because CoTC happens at the end of region 11. Finally $DNAp27SA_{l_i}$ refer to the transcription of the last part of 27SA2 after the cleavage. It concerns the regions from 1 to 10. The only species initially different from zero are the ones

Table 1. The list of reactions in the Dizzy model

$transcr_{l_i}$	$DNA_{l_i} \rightarrow DNA_{l_{i+1}} + El35S_{l_i}$,	rt; i=1,2,...,16
$transcr_{l_{17}}$	$DNA_{l_{17}} \rightarrow DNA_{l_{17}} + El35S_{l_{17}}$	rt;
$transcrp27SA_{l_i}$	$DNAp27SA_{l_i} \rightarrow DNAp27SA_{l_{i+1}} + El27SA2_{l_i}$,	rt; i=1,2,...9
$transcrp27SA_{l_{10}}$	$DNAp27SA_{l_{10}} \rightarrow El27SA2_{l_{10}}$,	rt;
$transcrCOTC_{l_{(i+10)}}$	$DNACoTC_{l_{(i+10)}} \rightarrow DNACoTC_{l_{(i+11)}} + DNAp27SA_{l_{10}} + 20S_{l_i}$,	rt; i=1,...6
$transcrCOTC_{l_{17}}$	$DNACoTC_{l_{17}} \rightarrow DNACoTC_{l_{17}} + DNAp27SA_{l_{10}} + 20S_{l_6}$,	rt;
$React3_{l_i}$	$El35S_{l_i} \rightarrow El20S_u + El27SA_{l_i}$	r3; i=1,2,..10
$React3_{l_i}$	$El35S_{l_i} \rightarrow El20Sl_{l_{(i-10)}} + El27SA2_{l_{10}}$	r3; i=11,12,..16
$React3_{l_{17}}$	$El35S_{l_{17}} \rightarrow El20Sl_{l_6} + El27SA2_{l_{10}}$,	r3;
$React4_{l_i}$	$El35S_{l_i} \rightarrow El23S_u + El27SB_{l_i}$,	r4; i=1,2,..10
$React4_{l_i}$	$El35S_{l_i} \rightarrow El23S_{l_{(i-10)}} + El27SB_{l_{10}}$,	r4; i=11,12,..17
$React5_{l_i}$	$El20S_{l_i} \rightarrow El18_{l_i}$,	r5; i=1,2,..6
$React5_u$	$El20S_u \rightarrow El18_u$,	r5;
$React6_{l_i}$	$El27SA2_{l_i} \rightarrow El27SB_{l_i}$,	r6; i=1,2,...,10
$React7_{l_i}$	$El27SB_{l_i} \rightarrow El5.8Su + El25S_{l_i}$,	r7; i=1,...,9
$React7_{l_{10}}$	$El27SB_{l_{10}} \rightarrow El5.8S_{l_i} + El25S_{l_9}$,	r7;
$React8_{l_1}$	$El7S_{l_1} \rightarrow El5.8_{l_1}$,	r8;
$React8_u$	$El7S_u \rightarrow El5.8_u$,	r8;

that refer to the rDNA regions where the nascent transcripts are close to the end of transcription at time t_0. These species may be considered soon "active":

$DNA_{l_1} = p1 * 100$;
$DNACoTC_{l_{11}} = p2 * 100$;
$DNAp27SA_{l_1} = p2 * 100$;

All others are considered null as they are initially "blocked" for transcription, only becoming active as the polymerase chains incorporating radioactive uracil move along the rDNA.

For the other species in the pathway, different statements are defined to represent the species at different levels of labelling. Each of these species are initially null as the labelling starts at time t_0. The unlabelled elements are not considered as they cannot be detected by the experiments.

Reactions. The reactions of the model are summarised in Table 1. They are all irreversible one-reactant reactions with mass action kinetics. The reaction names used in Dizzy correspond to the labels used in Figs. 2 and 3. The former three groups concern transcription, whereas the remaining ones describe the remaining reactions in the pathway with the addition of the labelling. It is worth noting that for each biological reaction in the two pathways we have as many reactions as the number of levels of labelling for that reactant.

The first kind of reactions is indicated by the name $transcr_{l_i}$ with $i = 1, 2, ..., 17$ and describes the usual transcription (the first reaction on the top in Fig. 2). For $i = 1, 2, 3, ...16$, the transcription of a nascent transcript initially

Table 2. Rate values for the reactions of the model

rate name	rt	r3	r4	r5	r6	r7	r8
value (s^{-1})	0.25	0.25	0.15	0.005	0.028	0.022	0.01

at level i results in pre-rRNA 35S at the level i of labelling, and $DNA_{l_{i+1}}$. Indeed after this transcription, the nascent transcripts initially at level $i+1$ are "activated". The last reaction describes the final situation in which we obtain fully-transcribed elements. The second and the third groups of reactions represent the translation with CoTC (the first reaction on the top in Fig. 3). The approach is similar to the one used in the usual transcription. The reactions labelled with $transcrp27SA_{l_i}$ indicate the transcription of the 27SA2 region in the case CoTC has happened before time $t = 0$ ($i = 1, 2, .., 9$) and in the case CoTC happens later ($i = 10$).

All the other reactions correspond to the remaining interactions shown in Figs. 2 and 3. These reactions represent either the cleavage of a pre-rRNA into two parts or in the degradation of some fragments (not represented explicitly in the model). The element $El35S_{l_i}$ is involved in two reactions: *React3i* and *React4i* for $i = 1, ..., 17$. These are the cleavage of 35S into $El20S_{l_i}$ and $El27SAS_{l_i}$ and into $El23S_{l_i}$ and $El27SB_{l_i}$, respectively. The reactions with names *React5i* for $i = 1, ..., 6$ and *React5u* are the transformation of the pre-rRNA 20S in the final rRNA $18S$ and, similarly, the reactions with name *React6i* for $i = 1, ..., 10$ stand for the transformation of 27SA2 into 27SB and *React81* and *React8u* for the transformation of $7S$ into the final rRNA $5.8S$. Finally the cleavage of $27SB$ into $25S$ and $7S$ is described by the reactions *React7i* for $i = 1, ..., 10$.

Rates. One of the main problems with the definition of the model has been the determination of the rates. Indeed the quantitative information and the data currently available from the experiments are neither complete nor numerous. We applied some *parameter estimation algorithms* discussed in [13,14] to our model. Broadly speaking, these algorithms are based on one objective function that measures the distance between the model and the experimental data. The parameters of the model are then adjusted to minimize the objective function. Among the different algorithms proposed in the literature we considered some global stochastic otimization methods, such as simulated annealing, genetic algorithms and evolutionary programming and some deterministic ones, such as steepest descent and Levenberg-Marquardt methods. In all cases either the approximation estimator errors were large or some errors occurred and the procedure stopped. In the future we aim to repeat the estimation procedure using many data-sets.

However, some information about the half life of the elements and the duration of the single reactions can be obtained from the literature and from some biological observations. Therefore we made use of these. A summary of the rates values used is reported in Table 2.

The transcription/labelling rate is assumed to be $rt = 0.25s^{-1}$. This value is calculated by considering that the transcription of each region happens in 10 s and an average of 2.5 transcripts for each region for each DNA is produced. We consider the number of transcripts equally distributed over the entire sequence. So we have: $rt = \frac{2.5}{10} = 0.25s^{-1}$. For all the other reactions, the constant rates are derived from the estimated duration of the reactions using the relation $rate = 1/time$. It is worth noting that the rate of a reaction does not depend on the labelling.

4 Validation and Experimentation

In this section we report the validation of the model and some analysis results. Gillespie's Direct Method is the simulation algorithm considered in this work.

4.1 Validation of the model

In order to validate the model, we simulate the model and compare the resulting curves with the known behaviours. In this study we are interested in reproducing results consistent with the knowledge of the model and not necessarily perfectly fitting the experimental data. Indeed at the moment we have only a few experimental data. However, they are sufficient to give an idea of the behaviour of the system. The variability of the experimental data is presumed to be relatively high, on average in the range of 10%. This does not change the overall shape of the curves.

Experimental data. In order to measure the amount of different elements, the pulse-labelling technique is used. Initially the cells are pulse-labelled with radioactive uracil. At different times, the pre-rRNA are extracted and isolated in a gel solution to be analysed. From this it is possible to derive the radioactive intensity of each element corresponding to a density "line". An example result of the experiment is shown in Fig. 5. The figure reports the variation of radioactive intensity for the different pre-rRNA and RNA with respect to time. The data derived from the pulse-labelling represent the radioactive density for some of the element in the pathway, normalised by the total number of uracil.

As the result of the simulation is given in terms of the amounts of species, we have to relate the intensity to the number of elements. The intensity depends on the number of the radioactive uracil and the proportion of uracil is approximately the same (30%) for each region. As a consequence of this, we can suppose that the intensity is proportional to the length of the labelled sequences. Given a pre-rRNA X composed of n_x levels of labelling, the intensity I of that element in a given time t is:

$$I \propto \sum_{k=1}^{n_x} \frac{k}{N} X_k(t) \tag{1}$$

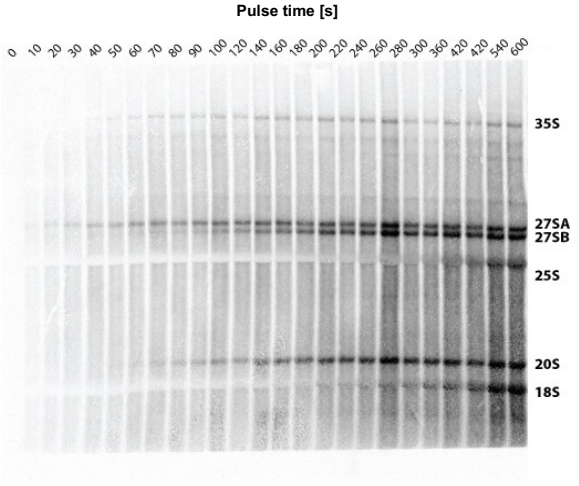


Fig. 5. Pulse-labelling experiment result: different pre-rRNAs are deposited at different levels within the gel according to their relative sizes (horizontal bands). As time progresses greater amounts of radioactive uracil are present and the intensity increases (vertical stripes moving from left to right).

where $N = 17$ is the total number of possible levels. The proportional factor is obtained by considering the experimental steady state values of all the components.

In Dizzy we can define new species representing the intensity of the elements detected in the experiments by using the expression 1.

Comparison. Figure 6 shows the time series obtained from experiments (on the left) and the results of the simulations (right). The experimental data refers to only one data-set, but this is expected to reproduce the correct biological behaviour, with the exception of some possible experimental error. The time for all the simulations is 500s, that is the time of the experimental data, and each is repeated for 100 runs. Each simulation lasts only a few minutes. In order to evaluate the goodness of the simulation we evaluated the confidence intervals for each simulation time point. In the case of a confidence coefficient $\alpha = 0.05$ we obtain that the confidence interval width is $3\% - 15\%$ of the steady state value of the respective species, with the exception of 35S, where the width can be larger. If we consider 1000 runs, the width of the confidence intervals decreases by an average of 50% with respect to the previous cases, but the simulation time increases to several minutes. Moreover, this variability in the simulation runs is in agreement with the variability intrinsic to the the biological model.

The simulation results are in agreement with the expected behaviours. The steady-state values obtained are comparable with the experimental data. There are however some discrepancies, in particular the drop around time t=300s in the 35S data and around time t=280s for 20S. These may be due to unknown

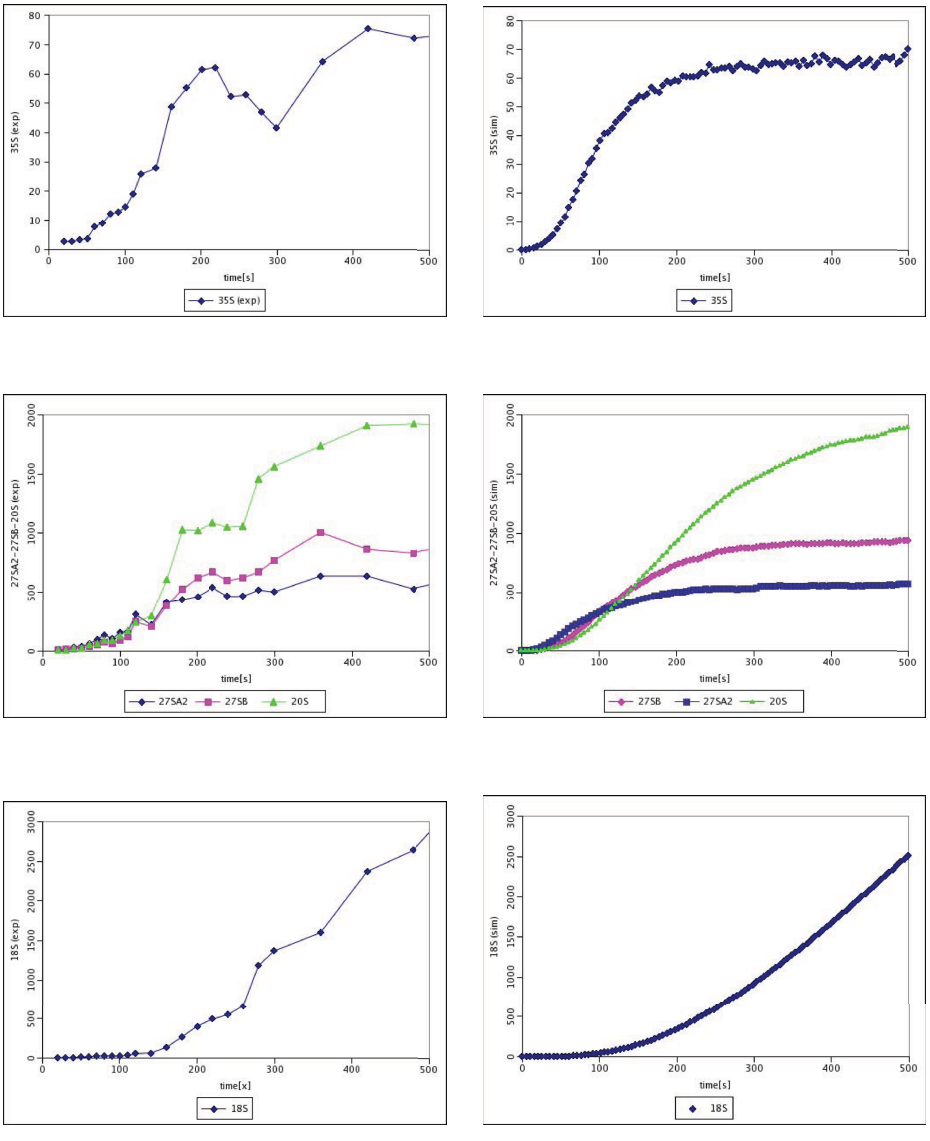


Fig. 6. Validation of the model. Experimental time-course data (one dataset, left) and the simulation result of species (right).

biological phenomena and further experiments are needed to investigate this aspect.

Concerning the first two plots at the top, 35S increases rapidly till its steady state. With respect to the experimental data, 35S increases faster in the first period, but in both cases the steady state is reached quite rapidly. In the case of 27SA2, 27SB and 20S the simulation results are in good agreement with the

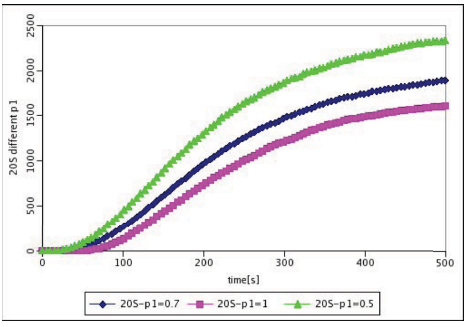


Fig. 7. Analysis results. Simulation of 20S for different values of the probabilities $p1$ and $p2$.

experimental data (curves in the middle). 27SA2 increases quickly and reaches the steady state quite rapidly. The element 27SB increases more slowly than 27SA2 in the first period but then it increases rapidly and reaches a steady state that is about twice the one of 27SA2. In the case of 20S the initial growth depends only on CoTC, but after some time there are also 20S obtained from 35S. Finally, 18S is very low in the first period, as it is obtained only after some steps in the pathway. After some time, it starts to accumulate, as expected.

4.2 Some analysis results

In the following we report some simulation results.

Influence of CoTC on the 20S formation. A first point to investigate is the influence of CoTC in the formation of the pre-rRNA 20S. In order to do this we plot 20S for different values of the frequencies $p1$ and $p2$. Some results are reported in Fig. 7. It shows the cases $p1 = 1$, $p1 = 0.7$ (the one chosen in the model) and $p1 = 0.5$. The change in the values of $p1$ (and consequently of $p2$) leads to a variation in the growth of 20S, especially in the first period. Indeed CoTC is responsible for the initial growth of 20S and in the case that we have no CoTC ($p1 = 1$) the curve initially increases very slowly. On the other hand the value of 35S is higher than in the case of $p1 = 0.7$. In the case of $p1 = 0.5$ the value of 20S increases faster and reaches a higher steady state, while the value of 35S is very low. For $p1 = 1$ and $p1 = 0.5$ the proportion of 20S with respect to 35S given in the experimental data is not consistent.

In order to see if the curves in Fig. 7 are statistically different from each other we consider the confidence interval for each time point. The confidence coefficient is fixed to 0.05. For most of the time points the interval confidences of the curves are not overlapping and so can be considered distinct.

These simulations seem to support the idea that the frequency of CoTC is non-null and it is a value close to the one chosen in the model.

Point of CoTC. We investigate by means of our model the influence of the point where CoTC happens on the growth of 20S and of 27S2. The results are reported in Fig. 8. The graph on the left concerns 20S and considers as point of CoTC the end of region 7, the end of region 13 and the end of the region 15. We see that the discrepancy between the different curves is minimal. However we can observe that the initial growth of 20S depends on the point of CoTC, in particular if CoTC happens towards the end of the transcription 20S increases slowly. Indeed by considering the confidence intervals for the points on the three curves, they overlap for most of the time points considered in the simulation. The graph on the right reports the same simulations for 27SA2. In this case the overlapping of the 3 curves is evident.

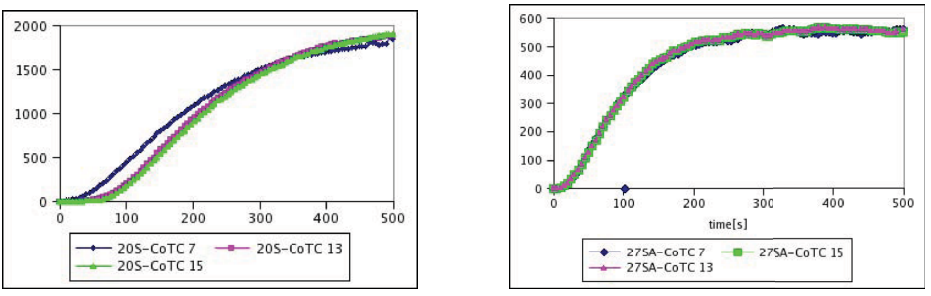


Fig. 8. Analysis results. Time series for 20S and 27SA2 for different points of CoTC.

5 Conclusions

In this paper we have presented a model describing the synthesis of pre-rRNA. In particular we focus on two aspects: the description of the process of labelling and on the choice between usual transcription and CoTC. Dizzy was chosen as the modelling and simulation framework for our study. The validation and the analysis have been made by means of stochastic simulation (Gillespie).

The model is able to reproduce the expected results and the simulations of the main elements are comparable with the experimental data. The fit is not exact but, as explained in the paper, there is currently little experimental data available. The experiments are quite complex and time-consuming and so the model can be used to make predictions that can then be checked by further experiments. Furthermore, as more experimental data becomes available we will be able to refine our model. Here we have shown the results of some preliminary studies. In particular we have used the model to investigate CoTC and its influence on the formation of 20S.

A major problem in the definition of the model has been the definition of the rates. In the present model they are derived from the literature or from experimental observations about the duration of reactions. One topic for future investigation will be the derivation of more precise rates, obtained by applying

parameter estimators [13] to a set of experimental data. Currently the experimental data we have are not enough to estimate the parameters by using these techniques. In the present work we have primarily been interested in reproducing of results comparable with the experimental data and our choice of parameters seems able to do that.

Another topic of future work will be to explore the present discrepancies between the experimental data and the model. In particular in the present experimental data there are some drops that are not captured by our model. Some further experiments are necessary to investigate the causes of these drops. If they are found to have biological causes we will modify our model in order to account for them.

Furthermore, we will try changing some of the hypotheses we have made. For example, our current model assumes that rates are constant, and that reactions follow a mass action kinetics. We could change this assumption and investigate if some alternative form of kinetic law is more appropriate for describing some of the processes in the biological model.

Last but not least, we aim to use our model to answer further biological questions about the synthesis pathway.

Acknowledgements. Federica Ciocchetta and Jane Hillston are supported by the EPSRC under the CODA project "Process Algebra Approaches for Collective Dynamics" (EP/c54370x/01 and ARF EP/c543696/01). Martin Kos is supported by European Commision (LSHG-CT-2005-518280) and David Tollervey is supported by The Wellcome Trust.

References

1. Nazar, R.N.: Ribosomal RNA processing and ribosome biogenesis in eukaryotes. IUBMB Life 56(8), 457–465 (2004)
2. Granneman, S., Baserga, S.J.: Crosstalk in gene expression: coupling and co-regulation of rDNA transcription, pre-ribosome assembly and pre-rRNA processing. Curr. Opin. Cell. Biol. 17(3), 281–286 (2005)
3. Singh, S., Yang, H.Y., Chen, M.Y., Yu, S.L.: A kinetic-dynamic model for regulatory RNA processing. J. Biotechnol. 127(3), 488–495 (2007)
4. Gilchrist, M.A., Wagner, A.: A model of protein translation including codon bias, nonsense errors, and ribosome recycling. J. Theor. Biol. 239(4), 417–434 (2006)
5. Cao, D., Parker, R.: Computational modeling of eukaryotic mRNA turnover. RNA 7(9), 1192–1212 (2001)
6. Miller, O.L., Beatty, B.R.: Portrait of a gene. J. cell. Physiol. 74, 225–232 (1969)
7. Ramsey, S., Orrell, D., Bolouri, H.: Dizzy: stochastic simulation of large-scale genetic regulatory networks. J. Bioinf. Comp. Biol. 3(2), 415–436 (2005)
8. Dizzy Home page, http://magnet.systemsbiology.net/software/Dizzy/
9. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 81(25), 2340–2361 (1977)
10. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. 104, 1876–1889 (2000)

11. Gillespie, D.T.: Approximate accelerated stochastic simulation of chemically reacting systems. J. Chem. Phys. 115, 1716–1733 (2001)
12. Gillespie, D.T., Petzold, L.R.: Improved Leap-Size Selection for Accelerated Stochastic Simulation. J. Chem. Phys. 119, 8229–8234 (2003)
13. Moles, C.G., Mendes, P., Banga, J.R.: Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods. Genome Res. 13, 2467–2474 (2003)
14. Mendes, P., Kell, D.B.: Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. Bioinformatics 14(10), 869–883 (1998)
15. Schweizer, E., MacKechnie, C., Halvorson, H.O.: The redundancy of ribosomal and transfer RNA genes in Saccharomyces cerevisiae. J. Mol. Biol. 40(2), 261–277 (1969)
16. Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis. Springer, Heidelberg (1993)

On the Analysis of Numerical Data Time Series in Temporal Logic

François Fages and Aurélien Rizk

Projet Contraintes, INRIA Rocquencourt,
BP105, 78153 Le Chesnay Cedex, France
Firstname.Lastname@inria.fr
http://contraintes.inria.fr

Abstract. Temporal logics and model-checking techniques have proved successful to respectively express biological properties of complex biochemical systems, and automatically verify their satisfaction in both qualitative and quantitative models. In this paper, we propose a finite time horizon model-checking algorithm for the existential fragment of LTL with numerical constraints over the reals, with the ability to compute the range of values of the real variables occurring in a formula that makes it true in a model. We illustrate this approach for the analysis of biological data time series, provide a set of biologically relevant patterns of formulas, and evaluate them on models of the cell cycle control and MAPK signal transduction.

1 Introduction

Temporal logics and model-checking techniques [1] have proved useful to respectively express biological properties of complex biochemical systems and automatically verify their satisfaction in both qualitative and quantitative models, i.e. in boolean [2,3,4], discrete [5,6], stochastic [7,8] and continuous models [9,10,3]. This approach relies on a logical paradigm for systems biology that consists in making the following identifications [11]:

$$\textit{biological model} = \textit{transition system}$$
$$\textit{biological property} = \textit{temporal logic formulae}$$
$$\textit{biological validation} = \textit{model-checking}$$

Having a formal language not only for describing models, i.e. transition systems by either process calculi [12,13,14,15,16], rules [2,17,18], Petri nets [19], etc..., but also for formalizing the biological properties of the system known from biological experiments under various conditions, opens a whole avenue of research for designing automated reasoning tools inspired from circuit and program verification to help the modeler [20]. However, the formalization of the biological properties as a specification in temporal logic remains a delicate task and a bottleneck of the method.

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 48–63, 2007.

In this paper, we investigate the use of this logical paradigm to analyze numerical data, and infer temporal logic specifications from experimental data time series. There has been work on the inference of correlations and positive and negative influences between entities from numerical data time series, especially for gene expression temporal data [21,22]. However to our knowledge, the inference of temporal logic formulae with real valued variables from numerical data time series is new.

In this paper, we generalize the finite time horizon model-checking algorithm described in [9] and recalled in the next section, to the existential fragment of LTL with numerical constraints over the reals. This first-order setting provides the ability to compute those instantiations of a formula that are true in a model, by giving the range of values of the real valued variables occurring in the formula for which it is true. The completeness of the algorithm is shown for the considered fragment of constraint-LTL in Sec. 3.

We illustrate the relevance of this approach to the analysis of biological data time series, by providing a set of biologically relevant patterns of formulas in Sec. 4, and by evaluating them on models of cell cycle control and of signal transduction in Sec. 5. We then conclude on the results achieved so far, their generality, and their use in on-going work.

2 Preliminaries on Model-Checking in Constraint-LTL over the Reals

2.1 Constraint-LTL over the Reals

The *Linear Time Logic* LTL is a temporal logic [1] that extends propositional or first-order logic with modal operators for qualifying when a formula is true in a tree of timed states, called a Kripke structure. The temporal operators are X ("next", for at the next time point), F ("finally", for at some time point in the future), G ("globally", for at all time points in the future), and U ("until"). These operators enjoy some simple duality properties, $\neg X\phi = X\neg\phi$, $\neg F\phi = G\neg\phi$, $\neg G\phi = F\neg\phi$, $\neg(\psi U\phi) = G\neg\phi \vee (\neg\phi U(\neg\phi \wedge \neg\psi))$, and $F\phi = \text{true}\ U\ \phi$.

A first-order version of LTL with constraints over the reals, called constraint-LTL, is used in Biocham [9] to express temporal properties about molecular concentrations. A similar approach is used in the Darpa BioSpice project [10]. Constraint-LTL considers first-order atomic formulae with equality, inequality and arithmetic operators ranging over real values of concentrations and of their derivatives. For instance `F([A]>10)` expresses that the concentration of `A` eventually gets above the threshold value 10. `G([A]+[B]<[C])` expresses that the concentration of C is always greater than the sum of the concentrations of A and B. Oscillation properties, abbreviated as `oscil(M,K)`, are defined as a change of sign of the derivative of M at least K times:

`F((d[M]/dt > 0) & F((d[M]/dt < 0) & F((d[M]/dt > 0)...)))` The abbreviated formula `oscil(M,K,V)` adds the constraint that the maximum concentration of M must be above the threshold V in at least K oscillations.

2.2 Model-Checking Algorithm

In an ODE model, and under the hypothesis that the initial state is completely defined, numerical integration methods (such as Runge-Kutta or Rosenbrock method for stiff systems) provide a discrete simulation trace. This trace constitutes a linear Kripke structure in which constraint-LTL formulae can be interpreted. Since constraints refer not only to concentrations, but also to their derivatives, we consider traces of the form

$$(< t_0, x_0, dx_0/dt, d^2x_0/dt^2 >, < t_1, x_1, dx_1/dt, d^2x_1/dt^2 >, ...)$$

where at each time point, t_i, the trace associates the concentration values x_i of the variables, and the values of their first and second derivatives dx_i/dt and d^2x_i/dt^2. This choice of derivatives is justified in section 4 as a facility for expressing positive or negative influences between entities. It is worth noting that in adaptive step size integration methods, the step size $t_{i+1} - t_i$ is not constant and is determined through an estimation of the error made by the discretization.

Algorithm 1 (trace-based constraint-LTL model-checking). *[9,10] Given an ODE model and a temporal property ϕ to verify within a finite time horizon,*

1. *compute a finite simulation trace;*
2. *label each trace point by the atomic sub-formulae of ϕ that are true at this point;*
3. *add sub-formulae of the form $F\phi$ (resp. $X\phi$) to the predecessors (resp. immediate predecessor) of a point labeled with ϕ;*
4. *add sub-formulae of the form $\phi_1\ U\ \phi_2$ to the points preceding a point labeled with ϕ_2 as long as ϕ_1 holds;*
5. *add sub-formulae of the form $G\phi$ to the last state if it is labeled by ϕ, and to the predecessors of the points labeled by $G\phi$ as long as ϕ holds.*
6. *return the time points labeled by ϕ.*

Note that being limited to finite traces, and since $G\phi = !F(!\phi)$, we chose to label the last state by $G\phi$ if it satisfies ϕ, just like if the trace was completed to the infinite by a loop on its last state. Note also that the notion of *next state* (operator X) refers to the state of the following time point in a discretized trace, and thus does not necessarily imply real time neighborhood. The rationale of this algorithm is that the numerical trace contains enough relevant points, and in particular those where the derivatives change abruptly, to correctly evaluate temporal logic formulae. This has been very well verified in practice with various examples of published mathematical models [9].

3 Formula Instantiation in ∃-Constraint-LTL over the Reals

3.1 The Existential Fragment ∃-Constraint-LTL

Here we consider the existential fragment of constraint-LTL, where real valued variables are allowed in the constraints, with an implicit existential quantification. More

precisely, the language of ∃-constraint-LTL formulae considered in this paper is defined by the following grammar :

$$
\begin{aligned}
Constraint-ltl = \\
&Atom \mid F(Constraint-ltl) \\
&\mid G(Constraint-ltl) \mid X(Constraint-ltl) \\
&\mid (Constraint-ltl)U(Constraint-ltl) \\
&\mid (Constraint-ltl)\ and\ (Constraint-ltl) \\
&\mid (Constraint-ltl)\ or\ (Constraint-ltl) \\
&\mid (Constraint-ltl) \Rightarrow (Constraint-ltl) \\
&\mid not\ (Constraint-ltl) \\
Atom = \\
&Value < Variable \mid Value \leq Variable \\
&\mid Value > Variable \mid Value \geq Variable \\
&\mid Value < Value \mid Value \leq Value \\
&\mid Value > Value \mid Value \geq Value \\
Value = \\
&float \mid [molecule] \mid d[molecule]/dt \mid d^2[molecule]/dt^2 \mid Time \\
&\mid Value + Value \mid Value - Value \mid -Value \mid Value \times Value \\
&\mid Value/Value \mid Value^{Value}
\end{aligned}
$$

By an obvious transformation, negations and implications can be eliminated, by propagating the negations down to the atomic constraints in the formula. From now on, we will assume that all ∃-constraint-LTL formulae are in negation free normal form.

3.2 Formula Instantiation Algorithm

Given a trace T representing a linear Kripke structure, and an ∃-constraint-LTL formula ϕ with n variables, the *formula instantiation problem*, $\exists \boldsymbol{v} \in \mathbb{R}^n\ (\phi(\boldsymbol{v}))$, is the problem of determining the valuation $\boldsymbol{v}$ of the variables for which the formula ϕ is true in T. In other words, we look for the domain of validity $\mathcal{D}_\phi \subset \mathbb{R}^n$ such that $T \models_{LTL} \forall \boldsymbol{v} \in \mathcal{D}_\phi\ (\phi(\boldsymbol{v}))$.

The domain of validity $\mathcal{D}_\phi$ of ϕ can be computed using an algorithm similar to the model-checking algorithm of section 2.2:

Algorithm 2 (trace-based ∃-constraint-LTL formula instantiation) *Given an ODE model and a temporal property ϕ with variables, to verify in a finite time horizon,*

1. *compute a finite simulation trace;*
2. *label each trace point by the atomic sub-formulae of* ϕ *and their domain of validity as follows :*
   - *for an atomic formula* ψ *without variables, label a time point* t_i *by* $(\psi, \mathcal{D}_\psi(t_i) = \mathbb{R}^n)$ *if* ψ *is true at time* t_i*, and* $(\psi, \mathcal{D}_\psi(t_i) = \varnothing)$ *otherwise;*
   - *for an atomic formula* $[A] \geq p$ *(that is, of the form value* $\geq$ *variable) label a time point* t_i *by* $([A] \geq p, \mathcal{D}_{[A]\geq p}(t_i))$ *where* $\mathcal{D}_{[A]\geq p}(t_i)$ *is the half-space of* $\mathbb{R}^n$ *defined by* $p \leq [A](t_i)$*;*
   - *proceed similarly for other atomic formulae containing variables;*
3. *starting from the end of the trace, label each time point* t_i *by the sub-formula* $F\psi$ *and its domain of validity* $\mathcal{D}_{F\psi}(t_i) = \mathcal{D}_{F\psi}(t_{i+1}) \cup \mathcal{D}_\psi(t_i)$*;*
4. *starting from the end of the trace, label each time point* t_i *by the sub-formula* $G\psi$ *and its domain of validity* $\mathcal{D}_{G\psi}(t_i) = \mathcal{D}_{G\psi}(t_{i+1}) \cap \mathcal{D}_\psi(t_i)$*;*
5. *starting from the end of the trace, label each time point* t_i *by the sub-formula* $\psi_1 U \psi_2$ *and its domain of validity* $\mathcal{D}_{\psi_1 U \psi_2}(t_i) = \mathcal{D}_{\psi_2}(t_i) \cup (\mathcal{D}_{\psi_1 U \psi_2}(t_{i+1}) \cap \mathcal{D}_{\psi_1}(t_i))$*;*
6. *label each time point* t_i *by the sub-formula* $X\psi$ *and its domain of validity* $\mathcal{D}_{X\psi}(t_i) = \mathcal{D}_\psi(t_{i+1})$*;*
7. *label each time point* t_i *by the sub-formula* ψ_1 *or* ψ_2 *and its domain of validity* $\mathcal{D}_{\psi_1 \text{ or } \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cup \mathcal{D}_{\psi_2}(t_i)$*;*
8. *label each time point* t_i *by the sub-formula* ψ_1 *and* ψ_2 *and its domain of validity* $\mathcal{D}_{\psi_1 \text{ and } \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)$*;*
9. *return the domain* $\mathcal{D}_\phi(t_i)$ *for all time points* t_i *where it is not empty.*

This algorithm enjoys a strong completeness theorem for the chosen fragment of constraints over the reals.

Theorem 1. *The instantiation algorithm is correct and complete: a valuation* $\boldsymbol{v}$ *makes* ϕ *true at time* t_i*,* $T, t_i \models_{LTL} (\phi(\boldsymbol{v}))$*, if and only if* $\boldsymbol{v}$ *is in the computed domain of* ϕ *at* t_i*,* $\boldsymbol{v} \in \mathcal{D}_\phi(t_i)$*.*

Proof. Let us prove inductively on the constraint-LTL formula structure that for any time t, any LTL formula ϕ and any instantiation $\boldsymbol{v}$ of the variables, if $\phi(\boldsymbol{v}, t_i)$ is true then $\boldsymbol{v} \in \mathcal{D}_\phi(t_i)$ and if $\boldsymbol{v} \in \mathcal{D}_\phi(t_i)$ then $\phi(\boldsymbol{v}, t_i)$ is true :

- Atomic constraint-LTL formulae considered are of the form $Value\ \mathcal{R}\ Variable$ or $Value\ \mathcal{R}\ Value$ where $Value$ is an evaluable arithmetic expression and $\mathcal{R}$ an inequality operator. For all these atomic formulae the algorithm returns the exact validity domain. For instance, formula $([A] \leq p)(t_i)$ is true if and only if p is greater or equal to $[A](t_i)$ and the validity domain returned is the half-space defined by $p \geq [A](t_i)$;
- ϕ_1 *and* ϕ_2 . By algorithm construction $\mathcal{D}_{\phi_1 \text{ and } \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)$ hence : $\boldsymbol{v} \in \mathcal{D}_{\phi_1 \text{ and } \phi_2}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_{\phi_1}(t_i)$ *and* $\boldsymbol{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\boldsymbol{v}, t_i)$ *and* $\phi_2(\boldsymbol{v}, t_i) \Leftrightarrow (\phi_1 \text{ and } \phi_2)(\boldsymbol{v}, t_i)$;
- ϕ_1 *or* ϕ_2 . By algorithm construction $\mathcal{D}_{\phi_1 \text{ or } \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cup \mathcal{D}_{\phi_2}(t_i)$ hence : $\boldsymbol{v} \in \mathcal{D}_{\phi_1 \text{ or } \phi_2}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_{\phi_1}(t_i)$ *or* $\boldsymbol{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\boldsymbol{v}, t_i)$ *or* $\phi_2(\boldsymbol{v}, t_i) \Leftrightarrow (\phi_1 \text{ or } \phi_2)(\boldsymbol{v}, t_i)$;

- $F(\phi)$. By algorithm construction $\mathcal{D}_{F(\phi)}(t_i) = \bigcup_{j \geq i}(\mathcal{D}_\phi(t_j))$ hence : $\boldsymbol{v} \in \mathcal{D}_{F(\phi)}(t_i) \Leftrightarrow \exists j \geq i, \boldsymbol{v} \in \mathcal{D}_\phi(t_j) \Leftrightarrow F(\phi)(\boldsymbol{v}, t_i)$;
- $G(\phi)$. By algorithm construction $\mathcal{D}_{F(\phi)}(t_i) = \bigcap_{j \geq i}(\mathcal{D}_\phi(t_j))$ hence : $\boldsymbol{v} \in \mathcal{D}_{G(\phi)}(t_i) \Leftrightarrow \forall j \geq i, \boldsymbol{v} \in \mathcal{D}_\phi(t_j) \Leftrightarrow G(\phi)(\boldsymbol{v}, t_i)$;
- $X(\phi)$. By algorithm construction $\mathcal{D}_{X(\phi)}(t_i) = \mathcal{D}_\phi(t_{i+1})$ hence : $\boldsymbol{v} \in \mathcal{D}_{X(\phi)}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_\phi(t_{i+1}) \Leftrightarrow X(\phi)(\boldsymbol{v}, t_i)$;
- $\phi_1 U \phi_2$. $\phi_1 U \phi_2(t_i)$ is true if and only if $\phi_2(t_i)$ is true or $\phi_1(t_i)$is true and $(\phi_1 U \phi_2)(t_{i+1})$ is true. Moreover by construction $\mathcal{D}_{(\phi_1 U \phi_2)}(t_i) = \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i))$ hence : $\boldsymbol{v} \in \mathcal{D}_{(\phi_1 U \phi_2)}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)) \Leftrightarrow (\phi_1 U \phi_2)(\boldsymbol{v}, t_i)$.

Now, let us define a box of $\mathbb{R}^n$ as a finite intersection of half-spaces and the size $S(\mathcal{D})$ of a domain as the minimum number of boxes the domain is made of. Note that for a one dimension domain $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ (resp. $\mathcal{D} = \mathcal{D}_1 \cap \mathcal{D}_2$), the maximum size is $S(\mathcal{D}_1) + S(\mathcal{D}_2)$ (resp. $\max(S(\mathcal{D}_1), S(\mathcal{D}_2))$).

Theorem 2. *In the worst case, the size of the validity domain of a LTL formula of size k on a trace of length n is n^{k^2}.*

Proof. Let us prove inductively that the size of the projection on one variable of the validity domain (i.e., the validity domain of a single variable) of a LTL formula of size k, is at most n^k. The size of the validity domain of an atomic formula is at most 1. The maximum size of a one dimension domain of a formula of size k is:

$$\begin{aligned}
\max(S(\mathcal{D}_{\phi_1}(t_i)), S(\mathcal{D}_{\phi_2}(t_i))) &\text{ for } \mathcal{D}_{\phi_1 \ and \ \phi_2}(t_i) \\
S(\mathcal{D}_{\phi_1}(t_i)) + S(\mathcal{D}_{\phi_2}(t_i)) &\text{ for } \mathcal{D}_{\phi_1 \ or \ \phi_2}(t_i) \\
\sum_{j \geq i} S(\mathcal{D}_\phi(t_j)) &\text{ for } \mathcal{D}_{F(\phi)(t_i)} \\
\max_{j \geq i} S(\mathcal{D}_\phi(t_j)) &\text{ for } \mathcal{D}_{G(\phi)(t_i)} \\
S(\mathcal{D}_\phi(t_{i+1})) &\text{ for } D_{X(\phi)}(t_i) \\
\max(S(\mathcal{D}_{\phi_1 U \phi_2}(t_{i+1})), S(\mathcal{D}_{\phi_1}(t_i))) + S(\mathcal{D}_{\phi_2}(t_i)) &\text{ for } D_{\phi_1 U \phi_2}(t_i)
\end{aligned}$$

In all these cases except operators F an U, the size of the domain is less than the sum of the domains' size of the subformulae at one time point, which entails a size smaller than $2n^{k-1}$. The U and F operators make a sum on all time points which entails a size of at most $n \times n^{k-1} = n^k$. Each projection's size of the validity domain is thus at most n^k. The size of the validity domain of a formula containing v variables is at most $(n^k)^v \leq (n^k)^k = n^{k^2}$.

The instantiation algorithm thus computes for each subformulae and each time point a validity domain of size at most n^{k^2}.

4 Biologically Relevant Patterns of ∃-Constraint-LTL Formulae

Temporal logic is sufficiently expressive to formalize a wide range of biological properties known from experiments under various conditions. The formula instantiation algorithm in ∃-constraint-LTL makes it possible to analyze concentration traces and obtain semi-quantitative information. In particular, a quantitative counterpart of the purely qualitative properties in propositional CTL studied in [3] can be expressed as follows, where variables are written using lowercase letters:

Reachability : `F([A]>=p)`, what threshold p species A attain in the trace ?

Checkpoints : `not (([A]<p1)U([B]>p2))`, for which thresholds $p1$ and $p2$ is it false that $[A]$ is lower than $p1$ until $[B]$ is above $p2$, i.e., for which $p1$ and $p2$ `[A] >= p1` is a checkpoint of `[B]>p2`?

Stability : `G([A]=<p1 & [A]>=p2)`, what is the range of values taken by $[A]$? This range can be looked for in some context given by a condition like in `G(Time>10 -> ([A]<p1 & [A]>p2))`.

Oscillation : `F((d([A])/dt>0 & [A]>v1) & (F((d([A])/dt<0 & [A]<v2))))`, what amplitude $(v1 - v2)$ is attained in at least one oscillation ? An oscillation is defined as the change of sign of the derivative. This formula can be extended for more oscillations and is abbreviated by `oscil(M,K,p)`. It states that M must have amplitude P in at least K oscillations. By applying the algorithm for each value of K, beginning with 1, we can find the number of oscillations in the trace and minimal amplitude P attained by K oscillations for any K.

Influence : `G(d[A]/dt>p1 -> d2[B]/dt2>=0)`, above which threshold does the derivative of A have an influence on B ? The influence is positive if a high value of `d[A]/dt` entails a positive second derivative of `[B]`. It is worth noticing that, as multiple species might influence B, this formula only indicates a correlation between the value of the derivative of A and the second derivative of B and gives no proof of direct influence.

5 Application to the Inference of Temporal Properties from Biological Time Series

5.1 Cell Cycle Data

In this section we present the application of the instantiation algorithm to the budding yeast cell cycle data. For the purpose of evaluation of the method, we do not use experimental data but simulation data obtained from the model of [23]. The application of the method to experimental data is discussed in section 5.3. Concentration traces are obtained by simulating the cell cycle control model in Biocham. Then, we try to recover relevant properties of the model by automatically analyzing the traces.

The reaction rules of the model are the following:

```
(1) _=>Cyclin.
(2) Cyclin+Cdc2~{p1}     => Cdc2-Cyclin~{p1,p2}
(3) Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
(4) Cdc2-Cyclin~{p1,p2} =[Cdc2-Cyclin~{p1}]=> Cdc2-Cyclin~{p1}
(5) Cdc2-Cyclin~{p1}     => Cyclin~{p1}+Cdc2
(6) Cyclin~{p1} =>_
(7) Cdc2         => Cdc2~{p1}
(8) Cdc2~{p1}    => Cdc2
```

Notations `~{p1}` and `~{p1,p2}` denote phosphorylated forms of a molecule. Figure 1 displays the obtained simulation traces for four species of this model.

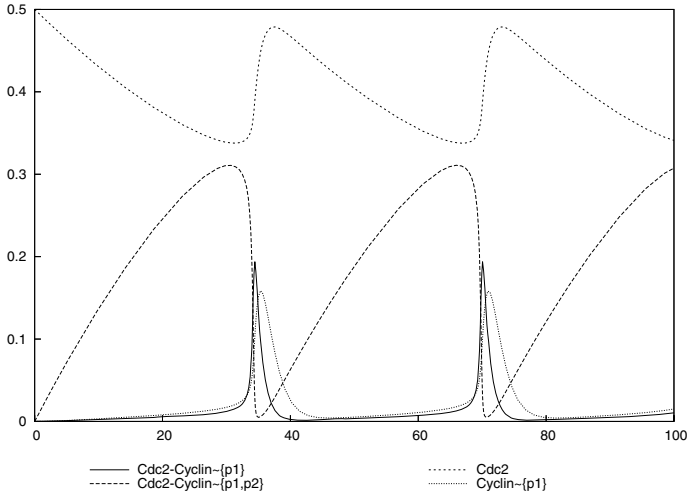


Fig. 1. Budding yeast cell cycle simulation trace over 100 time units made of 94 time points

Such traces are remarkably informative, however to automate reasoning on them, we propose to rely on constraint-LTL queries. For instance, a reachability query provides the maximum concentration attained by an entity:

```
biocham: trace_analyze(F([Cdc2-Cyclin~{p1}]>=v)).
[[v=<0.194]]
```

The result returned is a list of domains represented by lists of constraints on the variables, here a single domain is returned with a single constraint on v. In formulae like `F([Cdc2-Cyclin~{p1}]>=v)` where the variable only appears in inequalities of the form $Value \geq Variable$ or $Value > Variable$, the most relevant point of the domain is the highest value of `v` in the domain, i.e. its boundary. Its value is here `0.194`, the maximum concentration of `Cdc2-Cyclin~{p1}` in the

Table 1. Results for reachability (maximum value), stability (bottom and top values in the last third part of the trace) and amplitude of at least n oscillations

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
`Cdc2`	0.500	(0.338,0.479)	0.141	0.138
`Cdc2-Cyclin~{p1,p2}`	0.311	(0.005,0.310)	0.306	0.306
`Cdc2-Cyclin~{p1}`	0.194	(0.002,0.194)	0.192	0.192
`Cyclin~{p1}`	0.159	(0.004,0.158)	0.155	0.154

trace. Table 1 gives the maximum reachable values for the four species displayed in Figure 1.

For stability, let us find the range of values taken by `[Cdc2]` in the last third part of the trace:

```
biocham: trace_analyze(G(Time>66 -> ([Cdc2]=<v1 & [Cdc2]>=v2))).
[[v1>=0.479, v2=<0.338]]
```

The domain is defined by the conjunction of the two constraints `v1 >= 0.479` and `v2 =< 0.338`. These values are the maximum and minimum values attained by `[Cdc2]` in the last third part of the trace. The results for the other species are given in Table 1.

An oscillation query may compute several interval domains:

```
biocham: trace_analyze(oscil(Cdc2,1)).
[[v2>=0.338, v1=<0.479], [v2>=0.341, v1=<0.479]]
```

The result is the union of two boxes. In such domains, the most relevant point is not obvious. Here we look for the maximum amplitude $v1 - v2$. The maximum is obtained in the domain with $v1 - v2 = 0.479 - 0.338 = 0.141$. This result states that at least one oscillation of `Cdc2` has an amplitude greater or equal to 0.141. The number of oscillations is then incremented until obtaining an empty validity domain. It is obtained for `Cdc2` with the query `oscil(Cdc2,3)`, stating that there are only two oscillations of `Cdc2` in the trace.

The results for the other species are given in Table 1. Obtaining the amplitude of the oscillations is useful to distinguish between mixed amplitudes oscillations in the trace. For instance, in noisy data the amplitude can be used to count the number of oscillations regardless of small noise induced oscillations.

Whether `Cdc2-Cyclin~{p1,p2}` acts as a checkpoint for `Cdc2-Cyclin~{p1}` can be investigated with the following formula:

```
not([Cdc2-Cyclin~{p1,p2}]<v1 U [Cdc2-Cyclin~{p1}]>v2)
```

The resulting domain is a union of ten boxes. Interpreting it requires examining each box to find interesting points of the domain. Checkpoint queries are thus more delicate and perhaps not well suited for automatic analysis. In the example, the values $v1 = 0.311$ and $v2 = 0.014$ are in the domain, stating that `Cdc2-Cyclin~{p1,p2}` is not always less than 0.311 until `Cdc2-Cyclin~{p1}` exceeds 0.014. In other words `Cdc2-Cyclin~{p1,p2}` goes beyond 0.311 before

Cdc2-Cyclin~{p1} exceeds 0.014 pointing out that Cdc2-Cyclin~{p1,p2} is indeed a checkpoint.

Now, the influence of a molecule A on a molecule B is looked for with formula G(d[A]/dt>p1 -> d2[B]/dt2>0). The idea behind this formula is that if a species B appears only in a reaction rule of the form $A \rightarrow B$ with a mass action law kinetic, the following constraint-LTL formulae are true : $G(d[A]/dt > 0 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < 0 \Rightarrow d^2[B]/dt^2 < 0)$.

In a typical system each species concentration is the result of the combined effect of several other species. $\exists$-constraint-LTL formula search determines above which threshold the above formulae are true, i.e. validity domains of variables v1 and v2 in formulae $G(d[A]/dt > v1 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < v2 \Rightarrow d^2[B]/dt^2 < 0)$.

By comparing these domains to the range of values of $d[A]/dt$, a score $s \in [0, 1]$ is obtained indicating the influence of the derivative of [A] over the second derivative of [B]. More precisely, if the domain of validity is $v1 \geq 0$ it means that the formula is true for any positive value of $d[A]/dt$ resulting in a score 1. If the domain of validity is $v1 >= \frac{\max(d[A]/dt)}{2}$ it means that the formula is true for half of the positive values of $d[A]/dt$ resulting in a score 0.5. Table 2 gives influences scores computed by this method for species Cdc2 and Cdc2-Cyclin~{p1,p2}.

Table 2. Positive influence scores of all species on Cdc2 and Cdc2-Cyclin~{p1,p2}. Molecules appearing in rows (resp .columns) act as molecule A (resp. B) in formulae $G(d[A]/dt > v1 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < v2 \Rightarrow d^2[B]/dt^2 < 0)$ used to compute these scores.

Species	Cdc2	Cdc2-Cyclin~{p1,p2}
Cdc2	0.00	0.11
Cdc2~{p1}	0.01	0.12
Cyclin	0.00	**0.34**
Cdc2-Cyclin~{p1,p2}	0.00	0.02
Cdc2-Cyclin~{p1}	**0.90**	0.00
Cyclin~{p1}	0.50	0.09

According to the reaction rules, the only species having a positive influence on [Cdc2] is [Cdc2-Cyclin~{p1}] (reaction (5)). The influence scores returned correctly reflect this. The score obtained by Cyclin~{p1} is due to its closeness with [Cdc2-Cyclin~{p1,p2}] as it can be seen in the trace. These two species both have a concentration rise coinciding with [Cdc2] own concentration rise. Nevertheless, influence scores defined above enable to distinguish [Cdc2-Cyclin~{p1}] over [Cyclin~{p1}] as molecule having a positive influence on Cdc2.

According to the reaction rules, the two species having a positive influence on Cdc2-Cyclin~{p1,p2} are [Cyclin] and [Cdc2~{p1}] (reaction (2)). Notice that as more species influence Cdc2-Cyclin~{p1,p2} than Cdc2, it is harder to find correlations between single species and Cdc2-Cyclin~{p1,p2}. Therefore

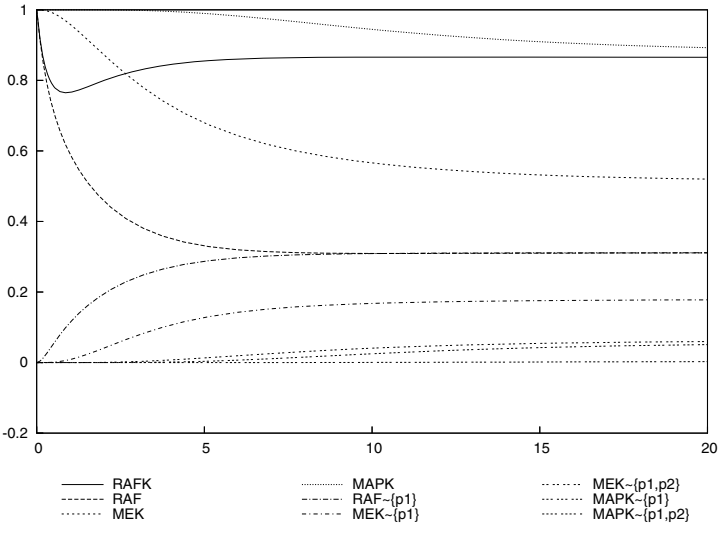


Fig. 2. MAPK model simulation trace over 20 time units made of 50 time points

overall influence scores are smaller in this case. In spite of this, the two species having the highest scores are the correct ones.

5.2 MAPK Signal Transduction Data

The MAPK signal transduction data model is used in the same way as the cell cycle model to evaluate the analysis method. Reaction rules used to simulate concentration traces, displayed in Figure 2 are given below. All reactions rules have mass action law kinetics.

```
(1) RAF + RAFK             <=> RAF-RAFK
(2) RAF~{p1} + RAFPH       <=> RAF~{p1}-RAFPH
(3) MEK~$P + RAF~{p1}      <=> MEK~$P-RAF~{p1}
   where p2 not in $P
(4) MEKPH + MEK~{p1}~$P    <=> MEK~{p1}~$P-MEKPH
(5) MAPK~$P + MEK~{p1,p2}  <=> MAPK~$P-MEK~{p1,p2}
   where p2 not in $P
(6) MAPKPH + MAPK~{p1}~$P  <=> MAPK~{p1}~$P-MAPKPH
(7) RAF-RAFK                => RAFK + RAF~{p1}
(8) RAF~{p1}-RAFPH          => RAF + RAFPH
(9) MEK~{p1}-RAF~{p1}       => MEK~{p1,p2} + RAF~{p1}
(10) MEK-RAF~{p1}           => MEK~{p1} + RAF~{p1}
(11) MEK~{p1}-MEKPH         => MEK + MEKPH
(12) MEK~{p1,p2}-MEKPH      => MEK~{p1} + MEKPH
(13) MAPK-MEK~{p1,p2}       => MAPK~{p1} + MEK~{p1,p2}
(14) MAPK~{p1}-MEK~{p1,p2}  => MAPK~{p1,p2} + MEK~{p1,p2}
(15) MAPK~{p1}-MAPKPH       => MAPK + MAPKPH
(16) MAPK~{p1,p2}-MAPKPH    => MAPK~{p1} + MAPKPH
```

Reachability, stability and oscillations queries results are given in Table 3. Stability queries return very small ranges of values in the last third part of the trace for most species indicating stable behaviors. For instance, `RAF` is inside the $[0.310, 0.311]$ interval in the last third part of the trace. There are no oscillations of the species except a very small one for `RAFK`.

Table 3. Results for Reachability (maximum value) and Stability (bottom and top values in the last third part of the trace)

Species	Reachability	Stability	Amplitude of at least n oscillations $n = 1$
`RAFK`	1	(0.865,0.866)	0.001
`RAF`	1	(0.310,0.311)	-
`MEK`	1	(0,519,0.537)	-
`MAPK}`	1	(0.891,0.916)	-
`RAF~{p1}}`	0.311	(0.311,0.311)	-
`MEK~{p1}`	0.178	(0.174,0.178)	-
`MEK~{p1,p2}`	0.060	(0.052,0.060)	-
`MAPK~{p1}`	0.052	(0.039,0.052)	-
`MAPK~{p1,)2}`	0.003	(0.001,0.003)	-

This model is made of a cascade of phosphorylation reactions. According to the reaction rules, `RAFK` acts as a kinase on `RAF` (reactions 1 and 7), `RAF` acts as a kinase on `MEK` (reactions 3, 9 and 10) and `MEK` acts as a kinase on `MAPK` (reactions 5,13 and 14).

We looked for positive influence of any species an all phosphorylated forms of `RAF`, `MEK` and `MAPK`. The highest score for `RAF~{p1}` is 0.96 and is attained by species `[RAF-RAFK]` while `[RAFK]` 's score is 0. This is consistent with the way phosphorylation reactions are written in the model, that is a complexation reaction and then a decomplexation-phosphorylation rule. Highest influence score for the phosphorylated form of `MEK` is correctly obtained for `[MEK-RAF~{p1}]`, while in the case of `[MAPK~{p1}]` the correct complex `[MAPK-MEK~{p1,p2}]` only gets the second highest score, and the situation is even more confused for `[MAPK~{p1,p2}]`.

Notice that lots of other species have relatively high influence score, which is no surprising given the similar shape of all curves in the trace. Nevertheless retaining only species having the highest scores as having positive influence, gives an overall good indication of the direct influences between species.

5.3 Experimental Data

Experimental data for measuring the evolution over time of gene expression levels or of protein concentrations, typically involve between 6 and 50 time points taken at regular intervals. Furthermore, experimental data are noisy, and it is not one trace but several ones that have to be analyzed in order to extract their

Table 4. Positive influence scores of all species on phosphorylated forms of `RAF`, `MEK` and `MAPK`

Species	RAF~{p1}	MEK~{p1}	MEK~{p1,p2}	MAPK~{p1}	MAPK~{p1,p2}
[RAFK]	0.00	0.11	0.46	**0.77**	**0.50**
[RAF]	0.00	0.00	0.00	0.20	0.02
[MEK]	0.26	0.00	0.00	0.00	0.00
[MAPK]	0.50	0.47	0.11	0.00	0.00
[MAPKPH]	0.50	0.49	0.20	0.01	0.00
[MEKPH]	0.48	0.06	0.00	0.00	0.00
[RAFPH]	0.14	0.00	0.00	0.00	0.00
[RAF-RAFK]	**0.96**	0.50	0.50	0.00	**0.50**
[RAFPH-RAF~{p1}]	0.00	0.22	0.47	0.42	**0.50**
[MEK-RAF~{p1}]	0.50	**0.79**	**0.66**	0.42	**0.50**
[MEK~{p1}-RAF~{p1}]	0.00	0.00	0.27	0.41	0.48
[MEKPH-MEK~{p1}]	0.00	0.00	0.34	0.45	0.49
[MEKPH-MEK~{p1,p2}]	0.00	0.00	0.00	0.60	0.34
[MAPK-MEK~{p1,p2}]	0.00	0.00	0.00	0.62	0.37
[MAPK~{p1}-MEK~{p1,p2}]	0.00	0.00	0.00	0.00	0.09
[MAPKPH-MAPK~{p1}]	0.00	0.00	0.00	0.00	0.19
[MAPKPH-MAPK~{p1,p2}]	0.00	0.00	0.00	0.00	0.00

significant features. The strategy here is thus to analyze the traces separately and retain the intersection set of their properties, or the most frequent ones only.

In order to evaluate the instantiation algorithm on similar experimental-like concentration traces, we extracted eleven equally spaced time points from the cell cycle simulation trace. The obtained trace is displayed in Figure 3.

Table 5. Results for reachability, stability and oscillation queries in experimental-like data

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
Cdc2	0.500	(0.341,0.441)	0.125	0.089
Cdc2-Cyclin~{p1,p2}	0.311	(0.031,0.308)	0.279	0.222
Cdc2-Cyclin~{p1}	0.194	(0.002,0.194)	0.192	0.012
Cyclin~{p1}	0.100	(0.005,0.100)	0.095	0.018

We applied on this trace the same queries than on the original simulated one, results are given in Tables 5 and 6. Oscillations properties are still obtained but with smaller amplitudes, because the peaks are missed in the sampling. For instance, `Cdc2-Cyclin~{p1}` has one oscillation of size 0.192 but two oscillations of size only greater than 0.012. This is a limit inherent to a low number of time points as the first peak of `Cdc2-Cyclin~{p1}` almost disappeared in this trace. Having a small number of time points also tends to give high self positive

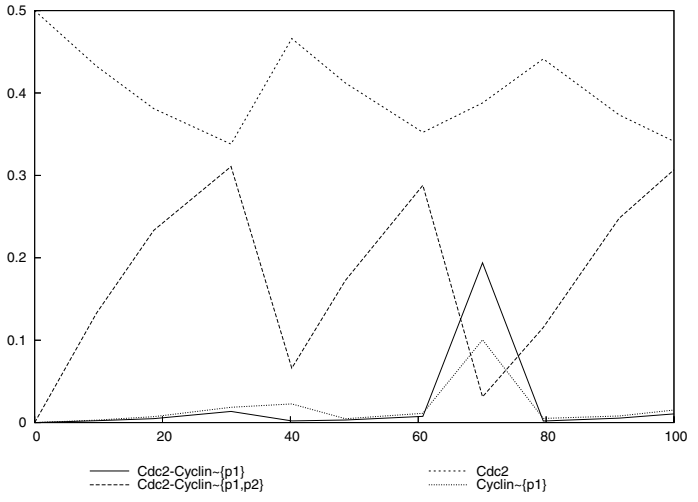


Fig. 3. Curve of concentrations every 10 units of time extracted from the cell cycle simulation trace

Table 6. Positive influence scores of all species on `Cdc2 and Cdc2-Cyclin~{p1,p2}`

Species	`Cdc2`	`Cdc2-Cyclin~{p1,p2}`
`Cdc2`	**0.59**	0.00
`Cdc2~{p1}`	**0.59**	0.00
`Cyclin`	0.00	**0.73**
`Cdc2-Cyclin~{p1,p2}`	0.00	0.59
`Cdc2-Cyclin~{p1}`	0.49	0.00
`Cyclin~{p1}`	0.48	0.00

influence scores but considering only highest scores except self influence still correctly determines the influence between species.

6 Conclusion

Considering the bottleneck of specifying in temporal logic with numerical constraints the biological properties of a system known from experiments, we have proposed an algorithm for inferring constraint-LTL formulae from numerical data time series. To this end, the finite time horizon model-checking algorithm described in [9] has been generalized to an instantiation algorithm in the existential fragment of LTL with numerical constraints over the reals. A strong completeness theorem stating that the ranges of real valued variables computed for a formula describe exactly the solution space, has been shown, together with a complexity bound in n^{k^2} on the representation of the domain, where n is the number of time points and k the size of the formula.

For the purpose of evaluating the method, we worked with time series generated from models by simulation, and considered one experimental-like time series extracted from the simulation trace in few time points taken at regular intervals of time. In the near future, we plan to apply the method to the analysis of experimental temporal data of FSH signaling proteins for designing a model of FSH signal transduction together with its temporal specification, and proceed similarly with cell cycle and circadian cycle data for cancer chronotherapies in the framework of the EU project Tempo[1].

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)
2. Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., Sönmez, M.K.: Pathway logic: Symbolic analysis of biological signaling. In: Proceedings of the seventh Pacific Symposium on Biocomputing, pp. 400–412 (2002)
3. Chabrier, N., Fages, F.: Symbolic model cheking of biochemical networks. In: Priami, C. (ed.) CMSB 2003. LNCS, vol. 2602, pp. 149–162. Springer, Heidelberg (2003)
4. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical interaction networks. Theoretical Computer Science 325, 25–44 (2004)
5. Bernot, G., Comet, J.P., Richard, A., Guespin, J.: A fruitful application of formal methods to biological regulatory networks: Extending thomas' asynchronous logical approach with temporal logic. Journal of Theoretical Biology 229, 339–347 (2004)
6. Batt, G., Bergamini, D., de Jong, H., Garavel, H., Mateescu, R.: Model checking genetic regulatory networks using gna and cadp. In: Graf, S., Mounier, L. (eds.) Model Checking Software. LNCS, vol. 2989, Springer, Heidelberg (2004)
7. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using the continuous time markow chains. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 44–67. Springer, Heidelberg (2006)
8. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. In: Priami, C. (ed.) CMSB 2006. LNCS (LNBI), vol. 4210, pp. 32–47. Springer, Heidelberg (2006)
9. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 68–94. Springer, Heidelberg (2006)
10. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics 38, 271–286 (2003)
11. Fages, F.: Temporal logic constraints in the biochemical abstract machine biocham (invited talk). In: Hill, P.M. (ed.) LOPSTR 2005. LNCS, vol. 3901, Springer, Heidelberg (2006)
12. Regev, A., Silverman, W., Shapiro, E.Y.: Representation and simulation of biochemical processes using the pi-calculus process algebra. In: Proceedings of the sixth Pacific Symposium of Biocomputing, pp. 459–470 (2001)

[1] `http://www.chrono-tempo.org/`

13. Cardelli, L.: Brane calculi - interactions of biological membranes. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 257–280. Springer, Heidelberg (2005)
14. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science 325, 141–167 (2004)
15. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science 325, 69–110 (2004)
16. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. Transactions on Computational Systems Biology. Special issue of BioConcur. (to appear, 2004)
17. Fages, F., Soliman, S., Chabrier-Rivier, N.: Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. Journal of Biological Physics and Chemistry 4, 64–73 (2004)
18. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. BioInformatics 22, 1805–1807 (2006)
19. Gilbert, D., Heiner, M., Lehrack, S.: A unifying framework for modelling and analysing biochemical pathways using petri nets. In: CMSB'07: Proceedings of the fifth international conference on Computational Methods in Systems Biology. LNCS, Springer, Heidelberg (2007)
20. Fages, F.: From syntax to semantics in systems biology - towards automated reasoning tools. Transactions on Computational Systems Biology IV 3939, 68–70 (2006)
21. Xu, R., Hu, X., II, D.C.W.: Inference of genetic regulatory networks from time series gene expression data. JCNN 2, 1215–1220 (2004)
22. Nachman, I., Regev, A., Friedman, N.: Inferring quantitative models of regulatory networks from expression data. In: ISMB/ECCB (Supplement of Bioinformatics), pp. 248–256 (2004)
23. Chen, K.C., Csikász-Nagy, A., Györffy, B., Val, J., Novàk, B., Tyson, J.J.: Kinetic analysis of a molecular model of the budding yeast cell cycle. Molecular Biology of the Cell 11, 391–396 (2000)

Context Sensitivity in Logical Modeling with Time Delays

Heike Siebert and Alexander Bockmayr

DFG Research Center Matheon,
Freie Universität Berlin, Arnimallee 3, D-14195 Berlin, Germany
siebert@mi.fu-berlin.de, bockmayr@mi.fu-berlin.de

Abstract. For modeling and analyzing regulatory networks based on qualitative information and possibly additional temporal constraints, approaches using hybrid automata can be very helpful. The formalism focussed on in this paper starts from the logical description developed by R. Thomas to capture network structure and qualitative behavior of a system. Using the framework of timed automata, the analysis of the dynamics can be refined by adding a continuous time evolution. This allows for the incorporation of data on time delays associated with specific processes. In general, structural aspects such as character and strength of interactions as well as time delays are context sensitive in the sense that they depend on the current state of the system. We propose an enhancement of the approach described above, integrating both structural and temporal context sensitivity.

1 Introduction

Logical modeling of bioregulatory networks started more than thirty years ago with the work of Sugita, Kauffman, Glass, and Thomas [10,4,3,12]. R. Thomas [12] introduced a logical formalism in the 1970s, which, over the years, has been further developed and successfully applied to different biological problems (see [13], [14] and references therein). Network components are modeled as discrete variables, the values of which correspond to the different expression levels of the component. In the simplest case, there are only two expression levels, 0 and 1, representing for instance whether or not a substance has surpassed some threshold concentration associated with a network interaction. A directed graph is used to represent interactions between the network components. Edges are labeled with signs corresponding to the character of the interaction, i. e., activating or inhibiting, and information on the thresholds associated with the interaction. In order to derive the dynamics of the system, parameters are introduced that take discrete values and determine for each network state the influence of enabled interactions on the system's behavior. To obtain a deterministic behavior, one would need sufficient data on the time delays associated with the different changes of expression level for all network components. In the classical Thomas formalism, the only assumption made is that all those time delays are different, resulting in a non-deterministic, asynchronous description of the dynamics. If more temporal data, e. g. in the form

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 64–79, 2007.

of time constraints, is available, we need a refined modelling approach allowing for the incorporation of such data. In [8] we proposed such a formalism based on the theory of timed automata. Here, each component is equipped with a clock used to evaluate conditions imposed on the time delays of this component during the evolution of the system.

In both approaches described so far the assumption is made that the information used for modeling is valid regardless of the state of the system. This assumption is often not met in reality. We have to deal with *context sensitivity* in the sense that characteristics of network interactions as well as time delays associated with changes in expression level may depend on the current state of the system. Whether some substance has an activating or inhibiting influence on the transcription of a gene, for example, may very well depend on the concentration of that substance. This is an example for *structural context sensitivity*. Moreover, the time delay for some component a to reach expression level 1 from 0 may be significantly shorter if the expression level change follows from activation by component b rather than from some influence exerted by component c, indicating *temporal context sensitivity*. Both kinds of phenomena may be of crucial importance for the system's functional purpose. Changes in the modeling formalism pertaining to both the logical modeling of the network structure and the parameter definition as well as the construction of the timed automaton model are necessary to capture the behavior of such context sensitive systems.

Specification of a model comprises three levels, allowing for a stepwise extension of the model in accordance with the available data. All the structural and discrete dynamical information about the network can be expressed in a context sensitive logical modeling framework based on the classical Thomas formalism. We develop this framework in Section 2. In a second step, discussed in Section 4, we show the translation of the context sensitive Thomas model into the framework of timed automata. At this stage we are able to incorporate information on time delays, using the properties of timed automata we introduced in Section 3. The resulting modeling approach constitutes a generalization of the one presented in [8]. As a final contribution we present a way to deal with the incorporation of context sensitive time delays in Section 5. We illustrate our methodology by examples, in particular the analysis of a regulatory network of bacteriophage λ, which we have implemented using the verification tool UPPAAL. We end the paper by discussing problems and perspectives of our approach

2 Context Sensitive Thomas Formalism

In this section we give a formal definition of a regulatory network based on the modeling approach of R. Thomas (see for example [13] and [14]). In contrast to Thomas' original approach our formalism allows for a rigorous description of a context sensitive system. The product of the cI gene in bacteriophage λ, for example, activates the cI gene by positive autoregulation. However, as the product concentration increases, the influence on its gene becomes inhibiting, thus preventing overexpression. In the classical Thomas formalism the interaction of cI

with itself is represented by a signed edge in the network graph, the sign characterizing the interaction as activating or inhibiting. In the described situation neither choice of sign reflects reality. It is still possible to choose the parameter values governing the system's dynamics to accommodate both activating and inhibiting effects, but that renders the edge sign meaningless. Since the edge signs play an important role in formulating general mathematical relations between motifs in the network graph and possible dynamical behavior, it seems important to keep them in the model of the network structure. Thus, we represent interactions capable of displaying activating as well as inhibiting effects by two edges, one negative, the other positive. In other words, while Thomas uses a directed graph to capture the structure of a network, we use a directed multigraph allowing for parallel edges. Consequently, we also need to alter Thomas' method of determining the system's dynamics from the graph. Multigraphs have already been used in [6] and [7], however, those graphs are derived from given discrete functions describing the system's behavior.

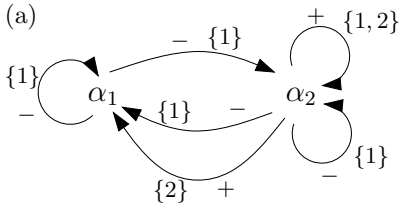
Definition 1. *An* interaction (multi-)graph *(or bioregulatory (multi-)graph) $\mathcal{I}$ is a labeled directed multigraph with*

- *vertex set $V := \{\alpha_1, \ldots, \alpha_n\}, n \in \mathbb{N}$, and*
- *edge (multi-)set $E \subseteq V \times V$. Let $\mathcal{T}(\alpha_j)$ be the set of all edges whose tail is α_j, and $\mathcal{H}(\alpha_i)$ the set of edges whose head is α_i. Each edge e from α_j to α_i is labeled with a sign $\varepsilon_e \in \{+, -\}$ and a set $M_e \subseteq \{1, \ldots, d_j\}$, $d_j \in \mathbb{N}$. For each sign $\varepsilon \in \{+, -\}$ there is at most one edge $\alpha_j \to \alpha_i$ labeled with ε.*

Let p_j be the maximal value of the union of all sets M_e with $e \in \mathcal{T}(\alpha_j)$. We call $\{0, \ldots, p_j\}$ the range *of α_j. For each $i \in \{1, \ldots, n\}$ we denote by $Pred(\alpha_i)$ the set of vertices α_j such that $\alpha_j \to \alpha_i$ is an edge in E.*

The vertices of this graph represent the components of the regulatory network, e. g. genes, the range of a vertex the different expression levels of the corresponding component affecting the behavior of the network. For example, if we only consider a component to be active or inactive, its range is $\{0, 1\}$. Thus, the vertices can be interpreted as variables that take values in the corresponding range. An edge e from α_j to α_i signifies that α_j influences α_i in a positive or negative way, depending on ε_e and provided that the current expression level of α_j is in M_e. If there is more than one edge leading from α_j to α_i, then the character of the interaction depends on the context. Figure 1 (a) shows a simple interaction graph comprising two vertices. The two different edges leading from α_2 to α_1 signify that α_2 has an inhibiting influence on α_1 if its expression level is 1. However, on the higher expression level 2 the influence becomes activating. Furthermore, there are two edges, e_3 and e_4, from α_2 to itself. The corresponding sets M_{e_3} and M_{e_4} intersect. It is not clear from the interaction graph alone, whether α_2 with expression level 1 inhibits or activates itself. The outcome depends on the interplay between the influence from α_2 on itself and the influence of α_1 on α_2. Thus, in order to determine the dynamical behavior of the system we need further information.

(a)



$e_1 := (\alpha_1, \alpha_1, -, \{1\})$,
$e_2 := (\alpha_1, \alpha_2, -, \{1\})$,
$e_3 := (\alpha_2, \alpha_2, +, \{1, 2\})$,
$e_4 := (\alpha_2, \alpha_2, -, \{1\})$,
$e_5 := (\alpha_2, \alpha_1, -, \{1\})$,
$e_6 = (\alpha_2, \alpha_1, +, \{2\})$

(b)

α_1	α_2	α_1'	α_2'	α_1'	α_2'
0	0	$K_{1,\{e_1,e_5\}}$	$K_{2,\{e_2,e_4\}}$	1	1
1	0	$K_{1,\{e_5\}}$	$K_{2,\{e_4\}}$	0	0
0	1	$K_{1,\{e_1\}}$	$K_{2,\{e_2,e_3\}}$	0	2
1	1	$K_{1,\emptyset}$	$K_{2,\{e_3\}}$	0	0
0	2	$K_{1,\{e_1,e_5,e_6\}}$	$K_{2,\{e_2,e_3,e_4\}}$	1	2
1	2	$K_{1,\{e_5,e_6\}}$	$K_{2,\{e_3,e_4\}}$	1	1

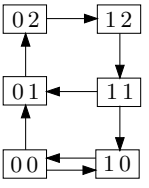


Fig. 1. In (a), interaction graph of a regulatory system comprising two components and six interactions. In (b), state table for general parameters with specific values, and the resulting state transition graph.

Definition 2. *Let $\mathcal{I}$ be an interaction graph. A* state *of the system described by $\mathcal{I}$ is a tuple $s \in S^n := \{0, \ldots, p_1\} \times \cdots \times \{0, \ldots, p_n\}$. The set of* resource edges *$R_i(s)$ of α_i in state s is the set*

$$\{e \in \mathcal{H}(\alpha_i) \mid (\varepsilon_e = + \wedge s_j \in M_e) \vee (\varepsilon_e = - \wedge s_j \notin M_e)\}.$$

Finally, given a set

$$K(\mathcal{I}) := \{K_{i,R_i(s)} \mid i \in \{1, \ldots, n\},\ s \in S^n\}$$

of (logical) parameters *$K_{i,R_i(s)}$, which take values in the range of α_i, we call the pair $(\mathcal{I}, K(\mathcal{I}))$ a* bioregulatory network.

In a given state s only the edges e with $s_{tail(e)} \in M_e$ represent active influences. The set of edge resources $R_i(s)$ contains all active positive edges and all inactive negative edges reaching α_i. That is, we interpret the absence of an inhibiting influence as activating influence. The value of the parameter $K_{i,R_i(s)}$ then indicates how the expression level of α_i will evolve. It will increase (resp. decrease) if the parameter value is greater (resp. smaller) than s_i. The expression level stays the same if both values are equal.

Typically, the set of resources $R_i(s)$, and with it the logical parameters, is defined as the set of predecessors of α_i (instead of edges reaching α_i) having an activating influence on α_i (see [2]). Since we allow for context sensitivity, knowledge of the current expression level of a predecessor of α_i is not enough to determine the character of the corresponding interaction. In the table given in Figure 1(b) we see in the second column the logical parameters determining

the state s' the system will evolve to from the state s given in the first column. The third column provides a specification of the parameter values. The resulting network behavior cannot be described in the classical Thomas formalism, since it is highly context sensitive. For example, we see that if α_2 has expression level 1, it activates itself in the absence of α_1, since $K_{2,\{e_2,e_3\}} = 2$. If α_1 is present, resulting in an effective inhibiting edge e_2, then α_2 inhibits itself, as indicated by the parameter choice $K_{2,\{e_3\}} = 0$. For an interaction graph without parallel edges, the notion of edge resources and vertex resources are equivalent.

The signs on the edges together with the sets M_e determine whether a component is an activator or an inhibitor of some other component in a given state. An activating influence, i. e., an effective activator or a non-effective inhibitor, cannot induce a decrease in expression level of the target component. This is reflected in the following parameter constraint:

$$\omega \subseteq \omega' \subseteq \mathcal{H}(\alpha_i) \Rightarrow K_{i,\omega} \leq K_{i,\omega'} \tag{1}$$

for all $i \in \{1, \ldots, n\}$. In the following we will always assume that this constraint is satisfied.

To conclude this section, we describe the dynamics of the system by means of a state transition graph.

Definition 3. *The* state transition graph $\mathcal{S}_N$ *corresponding to the bioregulatory network* $N = (\mathcal{I}, K(\mathcal{I}))$ *is a directed graph with vertex set* S^n*. There is an edge* $s \to s'$ *if there is* $i \in \{1, \ldots, n\}$ *such that* $s'_i = s_i + sgn(K_{i,R_i(s)} - s_i) \neq s_i$ *and* $s_j = s'_j$ *for all* $j \in \{1, \ldots, n\} \setminus \{i\}$*.*

To describe the dynamics of the system we use the so-called asynchronous update, i. e., a state differs from a successor state in one component only. If s is a state such that an evolution in more than one component is indicated, then there will be more than one successor of s. Note that s is a steady state if s has no outgoing edge. In Figure 1(b) we see the state transition graph corresponding to the state table also given in the figure.

3 Timed Automata

In this section we formally introduce timed automata. We mainly use the definitions and notations given in [1]. To introduce the concept of time in our system, we consider a set $C := \{c_1, \ldots, c_n\}$ of real variables that behave according to the differential equations $\dot{c}_i = 1$. These variables are called *clocks*. They progress synchronously and can be reset to zero under certain conditions. We define the set $\Phi(C)$ of *clock constraints* φ by the grammar

$$\varphi ::= c \leq q \,|\, c \geq q \,|\, c < q \,|\, c > q \,|\, \varphi_1 \wedge \varphi_2 \,,$$

where $c \in C$ and q is a rational constant.

A *clock interpretation* is a function $u : C \to \mathbb{R}_{\geq 0}$ from the set of clocks to the non-negative reals. For $\delta \in \mathbb{R}_{\geq 0}$, we denote by $u + \delta$ the clock interpretation

that maps each $c \in C$ to $u(c) + \delta$. For $Y \subseteq C$, we indicate by $u[Y := 0]$ the clock interpretation that maps $c \in Y$ to zero and agrees with u over $C \setminus Y$. A clock interpretation u satisfies a clock constraint φ if $\varphi(u) = true$. The set of all clock interpretations is denoted by $\mathbb{R}^C_{\geq 0}$.

Definition 4. *A* timed automaton *is a tuple* $(L, L^0, \Sigma, C, I, E)$*, where*

- L *is a finite set of* locations*,*
- $L^0 \subseteq L$ *is the set of* initial locations*,*
- Σ *is a finite set of* events *(or labels),*
- C *is a finite set of* clocks*,*
- $I : L \to \Phi(C)$ *is a mapping that labels each location with some clock constraint called the* invariant *of the location,*
- $E \subseteq L \times \Sigma \times \Phi(C) \times 2^C \times L$ *is a set of* switches*.*

A timed automaton can be represented as a directed graph with vertex set L. The vertices are labelled with the corresponding invariants and are marked as initial locations if they belong to L^0. The edges of the graph correspond to the switches and are labelled with an event, a clock constraint called *guard* specifying when the switch is enabled, and a subset of C comprising the clocks that are reset to zero with this switch. While switches are instantaneous, time may elapse in a location. To describe the dynamics of such an automaton formally, we use the notion of a transition system.

Definition 5. *Let A be a timed automaton. The* (labelled) transition system T_A *associated with A is a tuple $(Q, Q^0, \Gamma, \to)$, where Q is the set of states $(l, u) \in L \times \mathbb{R}^C_{\geq 0}$ such that u satisfies the invariant $I(l)$, Q^0 comprises the states $(l, u) \in Q$ where $l \in L^0$ and u ascribes the value zero to each clock, and $\Gamma := \Sigma \cup \mathbb{R}_{\geq 0}$. Moreover, $\to \subseteq Q \times \Gamma \times Q$ is defined as the set comprising*

- *transitions $(l, u) \xrightarrow{\delta} (l, u + \delta)$ for $\delta \in \mathbb{R}_{\geq 0}$ such that for all $0 \leq \delta' \leq \delta$ the clock interpretation $u + \delta'$ satisfies the invariant $I(l)$, and*
- *transitions $(l, u) \xrightarrow{a} (l', u[R := 0])$ for $a \in \Sigma$ such that there is a switch (l, a, φ, R, l') in E, u satisfies φ, and $u[R := 0]$ satisfies $I(l')$.*

The elements of $\to$ are called transitions*.*

The first kind of transition is a state change due to elapse of time, while the second one is due to a location-switch and is called *discrete*. Again we can visualize the object T_A as a directed graph with vertex set Q and edges corresponding to the transitions given by $\to$. Note, that by definition the set of states may be infinite and that the transition system is in general nondeterministic, i. e., a state may have more than one successor. Moreover, it is possible that a state is the source for edges labelled with a real value as well as for edges labelled with events. However, although every discrete transition corresponds to a switch in A, there may be switches in A that do not lead to a transition in T_A. That is due to the additional conditions placed on the clock interpretations.

Finally, we obtain a modified transition system by considering only the location vectors as states, dropping all transitions labelled with real values, but keeping every discrete transition of T_A. We call this the *discrete (or symbolic) transition system* of A.

4 Augmenting the Context Sensitive Thomas Formalism with Time Delays

We now present a generalization of the modeling approach using timed automata introduced in [8] suitable for regulatory networks displaying structural context sensitivity. Basically, we model each component of the system individually as a timed automaton, and then present a procedure to combine those elements to a timed automaton capturing the dynamical behavior of the system. This is done much in the same way a product automaton is derived from n timed automata (see [1]) and has been explained in detail in [8] and [9]. We illustrate the procedure and in particular the differences occurring due to the incorporation of context sensitivity using the example in Figure 1. Rigorous definitions will of course also be given for the alterations necessary in this more general approach.

4.1 Modeling the Components

We start modeling the components α_1 and α_2 of the system given in Figure 1 as timed automata A_1 and A_2.

Clocks. We use a single clock c_i for each component in order to measure the time needed for changes in expression level of that component. Even when introducing context sensitive time delays later on, we do not need more than one clock.

Locations. For each A_i we need a set of locations L_i. We introduce locations α_i^k for k in the range of α_i representing a situation where α_i maintains expression level k. They are called *regular* locations. Since we want to measure time delays corresponding to the increase or decrease of expression level, we furthermore define locations α_i^{k+} (resp. α_i^{k-}) for $k \in \{0, \ldots, p_i - 1\}$ (resp. $k \in \{1, \ldots, p_i\}$) indicating that the expression level is still k but is in the process of increasing (resp. decreasing). We call them *intermediate* locations. Lastly, we define $L_i^0 := \{\alpha_i^k ; k \in \{0, \ldots, p_i\}\}$. In Figure 2 the four locations of A_1 and the seven locations of A_2 are drawn as ellipses.

Invariants. The invariants of regular and intermediate locations are fundamentally different. Whether or not the component remains in a regular location does not depend on how much time has passed since it entered that location. In that sense regular locations are stable. This is reflected by assigning the invariant $c_i \geq 0$, which is true for every clock value, to every location α_i^k. In contrast, the intermediate locations are of transient character. The component will leave an intermediate location when the time needed for the corresponding expression level change has passed. We assign the location $\alpha_i^{k\varepsilon}$ the invariant $c_i \leq T_i^{k\varepsilon}$,

$\varepsilon \in \{+,-\}$, where $T_i^{k\varepsilon} \in \mathbb{Q}_{\geq 0}$ denotes the maximal time delay of the corresponding expression level change. In Figure 2 the invariant of each location is given in the line beneath the location name.

Switches and events. In the same way we use the invariants in the intermediate locations to include maximal time delays, we use the guards of the switches in E_i to introduce the minimal time delay $t_i^{k\varepsilon} \in \mathbb{Q}_{\geq 0}$ needed for an expression level change. For all $k \in \{0,\ldots,p_i-1\}$, we have $(\alpha_i^{k+}, a_i^{k+}, \varphi_i^{k+}, \{c_i\}, \alpha_i^{k+1}) \in E_i$, with $\varphi_i^{k+} = (c_i \geq t_i^{k+)})$, representing increase of expression level. Furthermore, for $l \in \{1,\ldots,p_i\}$, the switch $(\alpha_i^{l-}, a_i^{l-}, \varphi_i^{l-}, \{c_i\}, \alpha_i^{l-1})$ with $\varphi_i^{l-} = (c_i \geq t_i^{l-})$ belongs to E_i and represents expression level decrease. Every switch entails a reset of the component clock. The events $a_i^{k\varepsilon} \in \Sigma_i$ will be used later to identify location changes due to elapse of time, and thus correspond to the intermediate locations and the switches in E_i. We set $\Sigma_i := \{a_i^{k+}, a_i^{m-} ; k \in \{0,\ldots,p_i-1\}, m \in \{1,\ldots,p_i\}\}$.

The automaton A_1 has only two switches, A_2 only four, as can be seen in Figure 2. Thus it is clear that the dynamics of the system given in Figure 1 is not yet captured by the automata A_1 and A_2. This does not surprise since we have not incorporated the way both components interact with each other. In a next step we translate the information inherent in the interaction graph and the parameter values of the system in Figure 1 into conditions determining when a location change, independent of clock values, should occur in A_i. We call the resulting conditions *switch conditions*. Note that they can only be evaluated in the network context since they generally depend on the expression level of more than one component.

Switch conditions. The definition of the switch conditions deviates from the one given in [8] and [9], despite appearing similar due to notation similar to that in the earlier papers. However, we have to keep in mind that we use the context sensitive version of the Thomas formalism resulting in basic conceptual differences. Here, we give the new definition for the general situation of a network comprising n components, i. e., n automata $A_i = (L_i, L_i^0, \Sigma_i, C_i, I_i, E_i)$ defined as above.

To formulate the switch conditions, we need to know how to obtain from a location the expression level of the corresponding component. We use the function $\iota : \bigcup_{j\in\{1,\ldots,n\}} L_j \to \mathbb{N}_0$ that maps the locations α_j^k, α_j^{k+} and α_j^{k-} to k. Let $k \in \{1,\ldots,p_i-1\}$ and consider a location of A_i that represents expression level k. First we determine the resource edges (see Def. 2) that influence the behavior of A_i in this location. For every edge $e \in \mathcal{H}(\alpha_i)$ with tail α_j and l_j a location of A_j let

$$\lambda_i^e(l_j) := \begin{cases} \iota(l_j) \in M_e\,, & \varepsilon_{ij} = + \\ \iota(l_j) \notin M_e\,, & \varepsilon_{ij} = - \end{cases}, \qquad \overline{\lambda}_i^e(l_j) := \begin{cases} \iota(l_j) \notin M_e\,, & \varepsilon_{ij} = + \\ \iota(l_j) \in M_e\,, & \varepsilon_{ij} = - \end{cases}.$$

Thus, if $\lambda_i^e(l_j)$ evaluates to *true*, then e is a resource edge if the system is in location l_j of A_j (and thus α_j has expression level k). If the negation is true, e is no resource edge in location l_j. We are now interested in the sets of resource

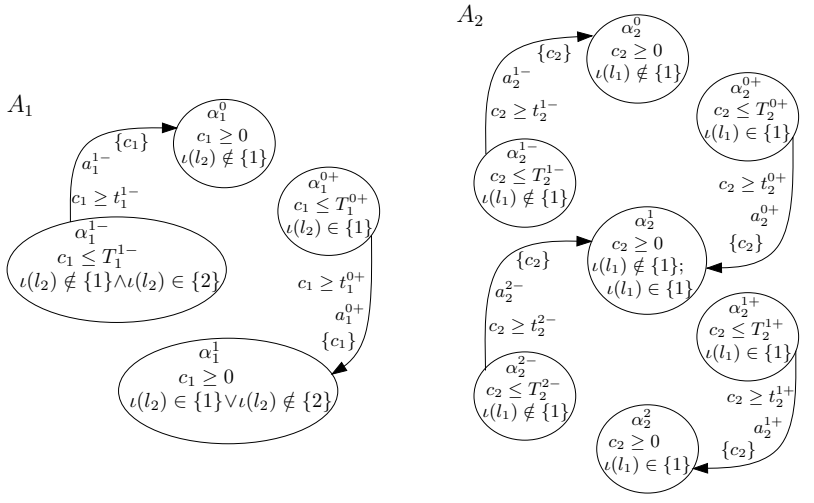


Fig. 2. Components A_1 and A_2 representing α_1 and α_2 in Figure 1

edges that effect a change in the expression level, and thus the location, of A_i. This information lies in the parameter values. Let $\omega_1, \ldots, \omega_{m_i^1}, \upsilon_1, \ldots, \upsilon_{m_i^2}$ be the subsets of $\mathcal{H}(\alpha_i)$ such that the parameter inequalities $K_{i,\omega_h} > k$ for all $h \in \{1, \ldots, m_i^1\}$ as well as $K_{i,\upsilon_h} < k$ for all $h \in \{1, \ldots, m_i^2\}$ hold. In our example, if we choose location α_2^1, i. e., $k = 1$, we can derive from the table in Figure 1(b) that $\omega_1 = \{e_2, e_3\}$ and $\omega_2 = \{e_2, e_3, e_4\}$, and $\upsilon_1 = \{e_3\}$ and $\upsilon_2 = \{e_4\}$.

Now we formulate conditions that check whether the system is in a state that provides the sets of resource edges necessary for an expression level change. Let $l \in L_1 \times \cdots \times L_n$, where l_i is the chosen location in A_i. Denote for each edge e by $t(e)$ the index of the tail of e, i. e., the tail of e is $\alpha_{t(e)}$. Then we define

$$\lambda_i^{\omega_h}(l) := \bigwedge_{e \in \omega_h} \lambda_i^e(l_{t(e)}) \qquad \text{and} \qquad \lambda_i^{\upsilon_h}(l) := \bigwedge_{e \in \mathcal{H}(\alpha_i) \setminus \upsilon_h} \overline{\lambda}_i^e(l_{t(e)}).$$

If $\lambda_i^{\omega_h}(l)$ is true for some ω_h, then an increase of expression level of gene α_i is indicated. If $\lambda_i^{\upsilon_h}(l)$ is satisfied for some υ_h, then the component will start the process of expression level decrease. In our example in location α_2^1 we obtain for ω_1 and υ_1 as determined above the conditions $\lambda_2^{\omega_1}(l) = (\iota(l_1) \notin \{1\}) \wedge (\iota(l_2) \in \{1, 2\})$ and $\lambda_2^{\upsilon_1}(l) = (\iota(l_1) \in \{1\}) \wedge (\iota(l_2) \in \{1\})$.

In order to induce a corresponding change in expression level, it is sufficient if the condition $\lambda_i^{\omega_h}(l)$ resp. $\lambda_i^{\upsilon_h}(l)$ holds for some ω_h resp. υ_h. Due to this observation we set

$$\Lambda_i^{k+}(l) := \bigvee_{h \in \{1, \ldots, m_i^1\}} \lambda_i^{\omega_h} \qquad \text{and} \qquad \Lambda_i^{k-}(l) := \bigvee_{h \in \{1, \ldots, m_i^2\}} \lambda_i^{\upsilon_h}.$$

We define Λ_i^{0+} and $\Lambda_i^{p_i-}$ accordingly.

Now, we assign all locations α_i^k, $k \in \{1, \ldots, p_i - 1\}$ the conditions Λ_i^{k+} and Λ_i^{k-}. The location α_i^0 resp. $\alpha_i^{p_i}$ is labelled with Λ_i^{0+} resp. $\Lambda_i^{p_i-}$ only, since the location represents the lowest resp. highest expression level possible. Furthermore, we want to check in an intermediate location whether the condition that led to the process of changing the expression level is still valid. If that is not the case, the system should not remain in that location. Thus, we associate with location α_i^{k+} the condition $\neg\Lambda_i^{k+}$ for all $k \in \{0, \ldots, p_i - 1\}$, and allot to location α_i^{k-} the condition $\neg\Lambda_i^{k-}$ for all $k \in \{1, \ldots, p_i\}$.

All the above considerations on how the switch conditions should influence the behavior of the system will be realized in the definition of the timed automaton representing the network dynamics.

Although the switch conditions look quite complicated, they often can be considerably simplified. Since condition (1) holds we can make the following observation. Whenever $\omega_{h_1} \subseteq \omega_{h_2}$ for sets ω_h, then $\lambda_i^{\omega_{h_1}}(l)$ is true if $\lambda_i^{\omega_{h_2}}(l)$ is true. Since condition (1) implies that $K_{i,\omega_{h_2}} \geq K_{i,\omega_{h_1}} > k$, we can delete condition $\lambda_i^{\omega_{h_2}}(l)$ from the expression $\Lambda_i^{k+}(l)$. Analogously, if $\upsilon_{h_1} \subseteq \upsilon_{h_2}$, we can delete the condition $\lambda_i^{\upsilon_{h_1}}(l)$ from the expression $\Lambda_i^{k-}(l)$. Furthermore, any inequality $\lambda_i^e(l_i)$ concerning the expression level $\iota(l_i)$ of the component A_i the inequality is associated with can be evaluated immediately. So, in the example we considered above, we have $\omega_1 \subseteq \omega_2$ and we can eliminate the condition $\lambda_2^{\omega_2}(l)$ from $\Lambda_2^{1+}(l) = \lambda_2^{\omega_1} \vee \lambda_2^{\omega_2}$, i. e., $\Lambda_2^{1+}(l) = \iota(l_1) \notin \{1\}$. For $\Lambda_2^{1-}(l)$ we have to consider both $\lambda_2^{\upsilon_1}(l)$ and $\lambda_2^{\upsilon_2}(l)$. However, we know that $\iota(l_2) = \iota(\alpha_2^1) = 1$, and thus we can simplify $\lambda_2^{\upsilon_1}(l) = (\iota(l_1) \in \{1\}) \wedge (\iota(l_2) \in \{1\}) = \iota(l_1) \in \{1\}$. The same reasoning shows that $\lambda_2^{\upsilon_2}(l) = (\iota(l_1) \in \{1\}) \wedge (\iota(l_2) \notin \{1, 2\})$ is always false in location α_2^1. Thus we can eliminate the second condition from Λ_2^{1-} and obtain $\Lambda_2^{1-} = \iota(l_1) \in \{1\}$. In Figure 2 the switch conditions for each location are listed below the invariant of the location.

4.2 Capturing the Network Dynamics

To obtain an automaton A from which we can derive the dynamics of the network, we have to combine the automata representing the individual components of the network. Again, we omit the details, which can be found in [9] and, with slightly different notation, in [8].

Locations, invariants and clocks. The set of locations of A is the product space of the location sets of the components A_i. Each location carries the conjunction of invariants of its components. A part of the timed automaton derived from the components A_1 and A_2 of our running example can be seen in Figure 3(a). Again, locations are shown as ellipses labeled with the location name and the corresponding invariant. The automaton A is equipped with the set of all component clocks.

Switches and events. Switches from the component automata persist, representing location changes that only affect the corresponding component of the location vector of A. Those edges are labeled with the events in Σ_i and depend only on the respective time delays. In our example automaton in Figure 3 two

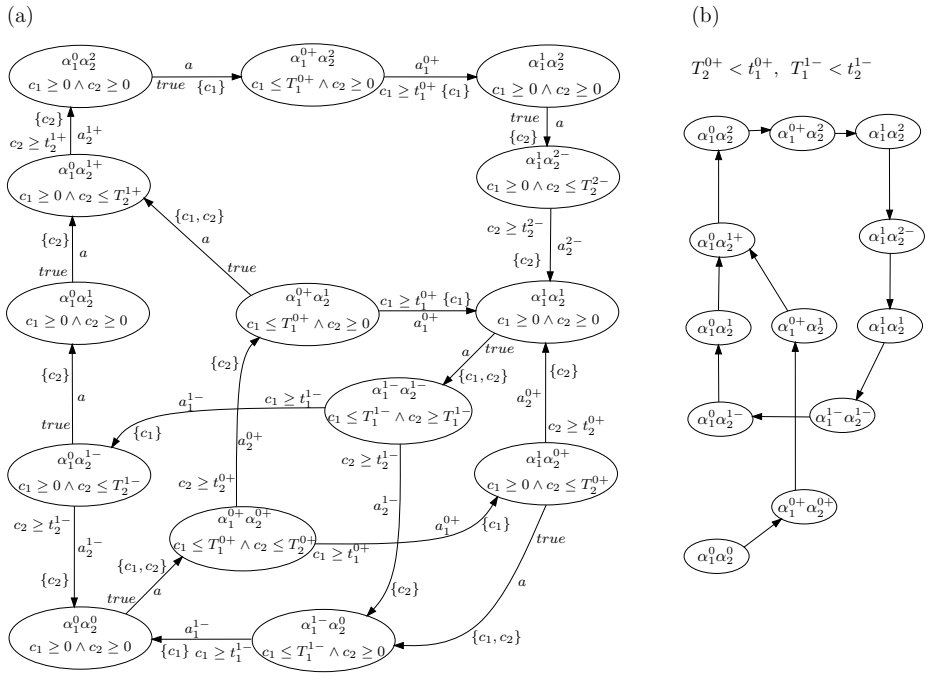


Fig. 3. In (a), part of the product automaton A derived from the components A_1 and A_2 given in Fig. 2. In (b), a path in the symbolic transition system respecting the time constraints given in the figure.

such edges leave the location $(\alpha_1^{0+}, \alpha_2^{0+})$. The one labeled with a_1^{0+} is inherited from A_1, the other one from A_2.

Furthermore, given a location l of A, we can evaluate all the switch conditions belonging to the component locations l_i. We define switches in A starting in l leading to a location l' differing from l in all those components l_i with true switch conditions. More precisely, if $l_i = \alpha_i^k$ is a regular location with true switch condition $\Lambda_i^{k\varepsilon}$ with expression level k and $\varepsilon \in \{+, -\}$, then $l'_i = \alpha_i^{k\varepsilon}$. This corresponds to the interpretation of a true switch condition as a set of resource edges effecting a process of expression level change. If $l_i = \alpha_i^{k\varepsilon}$ is an intermediate location with true switch condition, then $l'_i = \alpha_i^k$, since the true switch condition signifies that the current state of the system does not support the process of expression level change of α_i any longer. The switches resulting from the evaluation of the switch conditions are labeled with the event a and have no guard, or rather they are labeled with the guard *true*. When executing such a switch all the clocks of the components undergoing a location change are reset. In our example we can see such a switch starting in (α_1^0, α_2^0) leading to $(\alpha_1^{0+}, \alpha_2^{0+})$, since the switch conditions of both locations of the respective component automata given in Figure 2 are true.

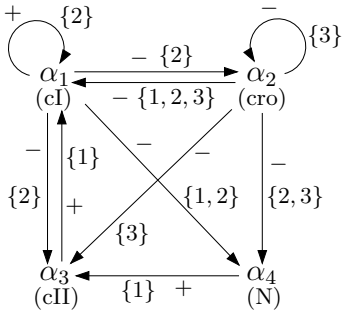
Transition system. The graph representing the automaton A does not represent the possible dynamical behavior of the system, since we have not yet evaluated the time constraints on switches and locations. Thus we have to derive the corresponding transition system. Basically, we follow the paths along the discrete location and switches in the graph representing A, if there exist clock values consistent with the time constraints we encounter along the way. That is, time may pass in locations as long as the maximal time delays in the invariants are not exceeded. Switches can be activated when the clock values are larger than the minimal time delays in the guards. Of course, every switch with the guard *true* can be executed regardless of the clock values. Executing switches is instantaneous and the reset commands have to be obeyed. We refine the resulting transition system in one aspect. Whenever the system is in a state allowing for the execution of a switch resulting from the evaluation of the switch conditions, i. e., a switch labeled with a, time is not allowed to elapse further in the corresponding location of A. Thus we ensure that the network interactions primarily determine the behavior of the system. For details see [8].

However, additional information about the time delays may lead to considerable refinement of the analysis of the system's dynamics. The state transition graph of our running example is strongly connected (see Figure 1(b)), prohibiting precise predictions of the system's behavior. This is also illustrated by Figure 3(a), which shows all the locations of the automaton A reachable from the location (α_1^0, α_2^0). However, given the additional information that the expression level increase from 0 to 1 is always faster for α_2, signifying for instance a higher production rate of a gene, and that the expression level decrease from 1 to 0 is always faster for α_1, representing for instance a high decay rate of some substance, we can obtain a much stronger understanding of the dynamics. Under those assumptions the system will reach, starting from (α_1^0, α_2^0), a cycle, that is a sustained oscillation, as shown in Figure 3(b).

5 Context Sensitivity of Time Delays

The current state of the system may not only influence the character of the network interactions but also the time delays associated with an expression level change. In this section we motivate why and illustrate how to incorporate context sensitivity of time delays into our modeling approach by considering the genetic switch of bacteriophage λ.

Phage λ is a virus that can act in two different ways upon infection of a bacterium. If they display the lytic response, the virus multiplies and lyses the cell. In other cases the viral DNA integrates into the bacterial chromosome, rendering the viral genome harmless for the so-called lysogenic bacterium. In [11] the authors propose a logical model of the genetic network underlying the behavior described above. It comprises the genes *cI*, *cro*, *cII* and *N*, the choice of parameter values and thresholds is based on experimental data. The resulting model is given in Figure 4. Here the inhibiting influence of *cI* on itself mentioned in Section 2 is not incorporated since it only takes place under conditions not of



$$
\begin{array}{lcl@{\qquad}lcl}
K_{1,\{e_{21}\}} & = & 2 & K_{2,\{e_{12}\}} & = & 2 \\
K_{1,\{e_{31}\}} & = & 2 & K_{2,\{e_{12},e_{22}\}} & = & 3 \\
K_{1,\{e_{11},e_{21}\}} & = & 2 & & & \\
K_{1,\{e_{11},e_{31}\}} & = & 2 & K_{3,\{e_{13},e_{23},e_{43}\}} & = & 1 \\
K_{1,\{e_{21},e_{31}\}} & = & 2 & & & \\
K_{1,\{e_{11},e_{21},e_{31}\}} & = & 2 & K_{4,\{e_{14},e_{24}\}} & = & 1
\end{array}
$$

Fig. 4. Model of the phage λ network. Edges are denoted by e_{ij} with α_i the tail and α_j the head of the edge. Only non-zero parameter values are given.

interest for the analysis of the behavioral switch (see again [11]). The lytic and lysogenic behavior can be identified in the state transition graph. The former is represented by the steady state $(2, 0, 0, 0)$ and the latter by a cycle comprising the states $(0, 2, 0, 0)$ and $(0, 3, 0, 0)$.

We have modeled this system as a timed automaton and implemented it in UPPAAL (see `http://www.uppaal.com`), a software that allows for verification and analysis via a model checking engine. In our model we exploited available temporal data to refine the results concerning the dynamical behavior (see [9] for details). However, certain aspects of the system are not correctly captured in the model as we will explain in the following. While genes *cI* and *cro* define the lysogenic and lytic states respectively, studies have shown the importance of *cII* in the switching process. The time delay values associated with accumulation and decay of the product of *cII* can be linked to environmental conditions such as richness of the medium (see [5]). Furthermore, it has been shown that the influence of *cII* leads to rapid synthesis of the *cI* product. When considering the parameter values given in Figure 4 we see that both the presence of *cII* and the absence of *cro* product are sufficient for *cI* to obtain its highest expression level. We lack the means to express the different time delays associated with the *cI* expression level change with respect to *cII* activity. However, since our modeling approach is highly suited for context sensitive systems, we can easily extend the model to capture the addressed properties.

Enhancing the Framework

The process of changing the expression level is represented by the intermediate location of the component automaton. If we want to associate different time delays with such a process we simply introduce two (or more) intermediate locations for the same process. We indicate the difference in the location name, e. g. with $\varepsilon \in \{+, -\}$ we denote by $\alpha_i^{k\varepsilon,s}$ (resp. $\alpha_i^{k\varepsilon,f}$) the slow (resp. fast) process of expression level change. For each location we choose a maximal time delay $T_i^{k\varepsilon,s}$ (resp. $T_i^{k\varepsilon,f}$) for the invariant. Each location is connected to the location α_i^l,

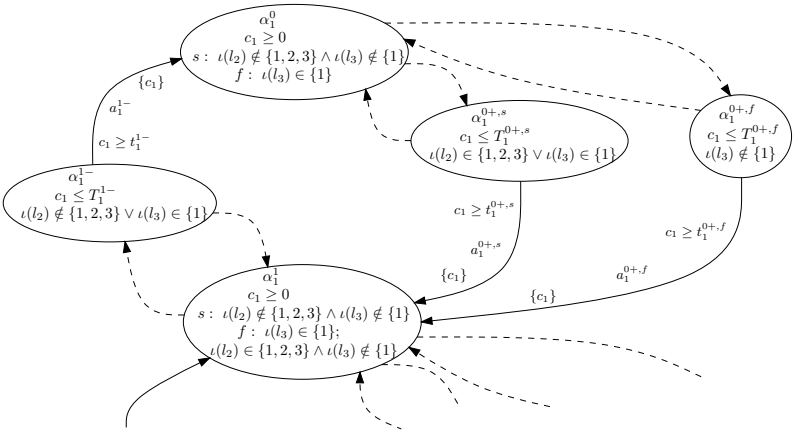


Fig. 5. Part of the timed automaton representing cI. Dashed arrows signify location changes due to evaluation of switch conditions.

with $l = k \pm 1$ depending on ε, via an edge labeled with the guard representing the corresponding minimal time delay $t_i^{k\varepsilon,s}$ (resp. $t_i^{k\varepsilon,f}$), an event $a_i^{k\varepsilon,s}$ (resp. $a_i^{k\varepsilon,f}$), and a reset for the component clock. In Figure 5 we see a part of the timed automaton representing cI with two intermediate locations representing the expression level change from 0 to 1.

In the context of the automaton representing the network, we decide via evaluation of the switch conditions if a system component executes a location change ending in an intermediate location. Now we have to classify the switch conditions such that we can distinguish between switches leading to different locations that represent the same process of expression level change. To do so we consider again the locations $\alpha_i^{k\varepsilon,s}$ and $\alpha_i^{k\varepsilon,f}$. If $\varepsilon = +$, we need to classify the switch conditions in location α_i^k. As a first step to formulating the switch conditions we determine the sets of resource edges $\omega_1, \ldots, \omega_{m_i^1} \subset \mathcal{H}(\alpha_i)$ that lead to the process of increasing the expression level k (see Section 4). Now we group the sets ω_h that lead to a fast change of expression level in a set Ω^f and the others in a set Ω^s. Then we derive a switch condition for the sets in Ω^f just as described in Section 4. If the system is in a state that the condition is true the component changes to the location $\alpha_i^{k+,f}$. We derive the switch condition for Ω^s also as described in Section 4, but additionally we demand that the switch condition of Ω^f is false. If this condition is met, then the component executes a location change to $\alpha_i^{k+,s}$. The switch conditions for both intermediate locations are, as usual, the negation of the switch conditions leading to the corresponding switch. If $\varepsilon = -$, we proceed analogously using the sets $\upsilon_1, \ldots, \upsilon_{m_i^2}$.

In order to model the more rapid expression level increase of cI in the presence of cII product, we first consider the sets of resource edges leading to the increase from 0 to 1. They are given in the left column of parameter values in Figure 4. A fast change is effected whenever e_{31} is a resource edge. Thus, we have $\{e_{31}\}$,

$\{e_{11}, e_{31}\}$, $\{e_{21}, e_{31}\}$ and $\{e_{11}, e_{21}, e_{31}\}$ in the set Ω^f. After the simplification of the switch condition as introduced in Section 4, we obtain $\iota(l_3) \in \{1\}$ as switch condition leading to fast expression level change. The switch condition for Ω^s is $\iota(l_2) \notin \{1, 2, 3\}$, and thus satisfying the condition $(\iota(l_2) \notin \{1, 2, 3, \}) \wedge (\iota(l_3) \notin \{1\})$ leads to the intermediate location $\alpha_1^{0+,s}$. The same considerations can be applied for the expression level change from 1 to 2. Part of the automaton is given in Figure 5. Here dashed arrows signify the location changes governed by the switch conditions in the network context.

Again we have implemented the model in UPPAAL and analyzed the behavior of the system. In a model with basically the same time delays for all location changes, we find that both the steady state representing the lysogenic behavior and the cycle representing lysis are reachable in the non-deterministic transition system. However, a faster expression level increase from 0 to 1 and a slower decrease from 1 to 0 of *cII* (both can be effected by a slower degradation rate) renders the cycle representing lysogeny unreachable. Thus, the system will always display lytic behavior. In contrast, smaller time delays for the expression level decrease and slower expression level increase of *cII* ensure that the system displays lysogenic behavior.

Regardless of the satisfactory result in the example considered above, the modeling of the context sensitive time delays may still be improved. Given the situation that the process of expression level change is nearly completed in the slow intermediate location, a change in the system's state that satisfies the conditions for a fast expression level change would lead to a repetition of the process of expression level change. All the progress made in the slow location would be lost. Although we could introduce switches leading directly from the slow to the fast intermediate location, we still have to decide which value to assign the clock upon execution of such a switch. The framework of timed automata only allows for two possibilities. Either the clock keeps its current value or it is reset to some constant. Obviously both choices do not reflect the desired behavior.

6 Perspectives

In this paper, we introduced a rigorous framework for the logical modeling of context sensitive systems. It extends our work on a hybrid formalism based on the classical Thomas approach and the theory of timed automata (see [8], [9]) in two directions. We are now not only able to model systems displaying context sensitivity regarding network interactions as described in Section 2, but can also deal with the context sensitivity of time delays, cf. Section 5. In many cases this allows for a more realistic representation of biological systems and a refined analysis of the resulting dynamics.

Generally, many interesting questions regarding modeling and dynamical analysis in this framework remain to be considered. For example, concepts like stability should now be phrased in a hybrid way, taking into account the time constraints associated with a certain behavior. This would allow for a more precise evaluation of asymptotical behavior, thus leading to more reliable predictions in case of system simulation as well as a better basis for model comparison.

In the current framework progress achieved in an intermediate location towards an expression level change, e. g. an increase in some substance concentration nearly up to the threshold, is completely negated if a location change signifying the abortion of the expression level change occurs. A more realistic representation should allow for a time delay, depending on how much time has passed in the intermediate location, associated with the loss of the progress made. This difficulty was already addressed at the end of the preceding section. It has to be considered whether the use of a more general class of hybrid automata would resolve this problem. However, powerful results concerning analysis and verification of models by means of model checking techniques exist in the theory of timed automata. Since effective methods to analyze large transition systems are needed in the context of biological systems, we should ensure that we do not loose advantages in that area. Rather, suitability and possibilities of applying model checking techniques for analyzing the behavior of biological networks need to be studied further.

References

1. Alur, R.: Timed Automata. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 8–22. Springer, Heidelberg (1999)
2. Bernot, G., Comet, J.-P., Richard, A., Guespin, J.: Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. J. Theor. Biol. 229, 339–347 (2004)
3. Glass, L., Kauffman, S.A.: The logical analysis of continuous, non-linear biochemical control networks. J. Theor. Biol. 39, 103–129 (1973)
4. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. 22, 437–467 (1969)
5. Oppenheim, A., Kobiler, O., Stavans, J., Court, D., Adhya, S.: Switches in bacteriophage lambda development. Annu. Rev. Genet. 39, 409–429 (2005)
6. Remy, É., Ruet, P., Thieffry, D.: Graphic requirements for multistability and attractive cycles in a boolean dynamical framework. Prépublication (2005)
7. Richard, A., Comet, J.-P.: Necessary conditions for multistationarity in discrete dynamical systems. Rapport de Recherche 123 (2005)
8. Siebert, H., Bockmayr, A.: Incorporating time delays into the logical analysis of gene regulatory networks. In: Priami, C. (ed.) CMSB 2006. LNCS (LNBI), vol. 4210, pp. 169–183. Springer, Heidelberg (2006)
9. Siebert, H., Bockmayr, A.: Temporal constraints in the logical analysis of regulatory networks. Matheon Preprint 385 (2007)
10. Sugita, M.: Functional analysis of chemical systems in vivo using a logical circuit equivalent. J. Theor. Biol. 1, 415–430 (1961)
11. Thieffry, D., Thomas, R.: Dynamical behaviour of biological regulatory networks - II Immunity control in bacteriophage lambda. Bull. Math. Biol. 57, 277–297 (1995)
12. Thomas, R.: Boolean formalisation of genetic control circuits. J. Theor. Biol. 42, 565–583 (1973)
13. Thomas, R., d'Ari, R.: Biological Feedback. CRC Press, Boca Raton, USA (1990)
14. Thomas, R., Kaufman, M.: Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits. Chaos 11, 180–195 (2001)

Stochastic Simulation of Biological Systems with Dynamical Compartment Structure

Cristian Versari and Nadia Busi

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
{versari,busi}@cs.unibo.it

Abstract. The Gillespie stochastic simulation algorithm represents one of the main physical abstractions exploited for the simulation of biological systems modeled by means of concurrent calculi. While the faithful modelling of bio-systems often requires multi-compartment semantics, the original Gillespie algorithm considers only one fixed-size volume. In this paper we introduce an extended formalisation of the above algorithm which preserves the original model but allows the stochastic simulation in presence of multiple compartments with dynamical structure and variable sizes. The presented algorithm can be then used as basis for simulating systems expressed in an extended version of the stochastic π-Calculus, the Sπ@ language, obtained by means of polyadic synchronisation. Despite of its conservativeness, Sπ@ is showed to allow flexible modelling of multiple compartments with dynamical structure and to provide increased biological faithfulness.

1 Introduction

The Gillespie stochastic simulation algorithm (SSA for short) represents one of the main physical abstractions exploited for the simulation of discrete, stochastic processes which model chemical and biological systems [1,2,3,4]. a single adiabatic compartment of fixed volume, containing a uniform mixture of chemical species interacting and producing new chemical reactants. While the abstraction of single, adiabatic, fixed-volume compartment is well suited for an isolated, gas-phase system as assumed in [5], the simulation of biological systems requires to enrich the model with some non-trivial details. As denoted by many recent approaches (e.g. [3,6,7]) the multi-compartment structure appears as a natural and indispensable requirement for the faithful representation of biological systems.

The most intuitive way to adapt the SSA to multiple compartments [8] is to consider each compartment as an isolated system evolving in parallel with the others, so that each compartment is implemented as an instance of the SSA. Even if correct, the above approach is based on strong assumptions which severely limit its application.

In fact, the distinctive behaviour of biological systems is the continuous interaction of their constituting elements. A compartmentalised model of a bio-systems cannot dispense with the capability of interaction between adjacent

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 80–95, 2007.

compartments by means of interaction of their respective elements or exchange of material: a refined model should also be able of representing a dynamical compartment structure. When the simulation of compartments is obtained by independent instances of the SSA under its original assumptions, the interaction between compartments must obey the hypothesis of volume invariance but, for a liquid-state system like biological ones usually are, this means that the exchanged elements must be inappreciable with respect to the dimension of the whole compartments. This approximation can be tolerated until very few molecules walk through during the time of the simulation, but becomes unreasonable when the aim is the modelling of compartments with dynamical structure (i.e. the system is subject to complex structural changes like for example the movement, merging, splitting of compartments), a necessary step towards a satisfactory representation of biological systems.

Besides the limit of static compartment structure, the above approach presents another relevant drawback. Consider the system in Fig. 1 (a), composed of two compartments with different volume. According to the model of the SSA, the molecules m_1 and m_2 have different probability of collision when floating in compartment C_1 instead of C_2. In fact, since C_2 is characterised by a larger volume, the probability to collide is lower than in C_1. Since the SSA requires to specify the reaction rate (more precisely, the "reaction probability per unit time") for each reaction, the same kind of reaction (like for example the one between m_1 and m_2) is characterised by a different rate for each compartment. In other words, in presence of K compartments, the same reaction rule (performed by the same kind of reactants, with the same physical properties) will have K different rates depending on the compartment the reaction is localised into, hence K different representations, one for each instance of the SSA.

In this paper we follow a different approach to the extension of the SSA to the multi-compartment model: here we show that by introducing in the model the informations pertaining the volumes, the SSA can be transparently adapted to the exact representation of multiple compartments. Furthermore, we show that the extended model is consistent with the kinetic hypotheses of the original one, but closer to the biological scenario and simpler w.r.t. to the approach discussed above, since allows to give unique description of elements and reactions while keeping consistent their rates in function of the enclosing compartment.

Then we show how the extended SSA can be smoothly used as basis for the simulation of systems modeled in a stochastic version of the π-Calculus added with polyadic synchronisation. This extension, we called Sπ@, exhibits a significant increase of expressiveness and faithfulness towards biological modellings, despite of its conservativeness w.r.t. both the π-Calculus and the original Gillespie model.

We consider the reader familiar with the Gillespie algorithm [5] and the SpiM simulator [4].

Structure of the paper. In Sect. 2 the extension of the original Gillespie stochastic model to the multi-compartment context is presented. In Sect. 3 the Sπ@ language is formalised and the multi-compartment version of the SSA is coded.

In Sect. 4 some chemical and biological modelling examples are shown and in Sect. 6 the pros and cons of the approach and future improvements are discussed.

2 Multi-Compartment Gillespie Stochastic Model

The original SSA considers a gas-phase system of fixed volume V, filled (in the simplest case) with two (distinct) molecular species S_1 and S_2 which can undergo the reaction R_1 (that is react through the *reaction channel* R_1). By simple kinetic deductions, the probability $c\ \mathrm{d}t$ of collision of two molecules m_1 and m_2 (modeled as hard spheres of radii d_1 and d_2) in the infinitesimal time interval $(t, t + \mathrm{d}t)$ is calculated as

$$c\ \mathrm{d}t = \overline{\mathrm{d}V_{coll}/V} = V^{-1}\pi d_{12}^2 \overline{v_{12}}\ \mathrm{d}t$$

where $\mathrm{d}V_{coll}$ is the average "collision volume", $d_{12} = d_1 + d_2$ and $\overline{v_{12}}$ is the average relative velocity of the two molecules. In the same way, the probability $c_1\ \mathrm{d}t$ of *reactive collision* of two molecules can be calculated as

$$c_1\ \mathrm{d}t = c\, c_r\ \mathrm{d}t = V^{-1}\pi d_{12}^2 \overline{v_{12}}\, c_r\, \mathrm{d}t$$

where c_r represents the probability that a given collision is actually reactive. In general, there exists a constant r which depends only on the physical properties of the two molecules and the temperature of the system, such that

$$c_1\ \mathrm{d}t = V^{-1}\, r\ \mathrm{d}t$$

Furthermore, if X_1 and X_2 are the number of molecules of type S_1 and S_2 respectively, the probability $a_1\ \mathrm{d}t$ that an R_1 reaction will occur somewhere inside V in the infinitesimal time interval $(t, t + \mathrm{d}t)$ is

$$a_1\ \mathrm{d}t = X_1 X_2 c_1\ \mathrm{d}t = V^{-1} X_1 X_2 r\ \mathrm{d}t \tag{1}$$

The value a_1 captures all the informations pertaining the *concentration* of the reactants S_1 and S_2. In the case of a single compartment of fixed volume, a_1 depends only on the quantity of reactants inside V, while in the case of multiple compartments or variable volume, expression (1) allows to calculate the effective reaction rates in function of the number of involved molecules inside each compartment and the volume of the compartment itself.

The effect of compartments is to *separate* the enclosed elements, that is to *prevent the interaction between elements placed in different compartments*. If m_{11}, m_{21} are two molecules of species S_1 and S_2 inside compartment C_1, and m_{12}, m_{22} are two molecules of species S_1 and S_2 inside compartment C_2, even if m_{11} may interact with m_{22} by an R_1 reaction, their collision is prevented because of the separation granted by compartment boundaries. Even if the pairs m_{11}, m_{21} and m_{12}, m_{22} may undergo the same reaction of type R_1, the reaction R_1 of elements inside compartment C_1 is completely independent of the

same reaction R_1 of elements inside compartment C_2. Therefore, in a multi-compartment environment each *reaction channel* should be denoted not only by the *type* of reaction but also by the *compartment* the reaction happens in.

By following these intuitions, the SSA can be transparently adapted to simulate the multi-compartment model (we call multi-compartment Gillespie model, MCGM for short) without affecting its original kinetic hypotheses.

If $S_1, \ldots, S_N$ are the chemical species and $R_1, \ldots, R_M$ are the reaction types, $X_{11}, \ldots, X_{NK}$ are the number of molecules of each of the N species inside each of the K compartments, $R_{11}, \ldots, R_{MK}$ are the reaction channels (in the number of $M \cdot K$, one for each reaction type inside each compartment) then we can introduce, in place of $\mathrm{P}(\tau, \mu)\,\mathrm{d}\tau$ [5], the expression $\mathrm{P}(\tau, (\mu, c))\,\mathrm{d}\tau$ which denotes *the probability that, given the state* $(X_{11}, \ldots, X_{NK})$ *at time* t*, the next reaction in* V *will occur in the infinitesimal time interval* $(t+\tau, t+\tau+\mathrm{d}\tau)$*, and will be a reaction through the channel* $R_{\mu c}$. Then, according to expression (1), we define the *propensity function* of the reaction $R_{\mu c}$:

$$a_{\mu c}\,\mathrm{d}t = V_c^{-1} h_{\mu c} r_\mu\,\mathrm{d}t \tag{2}$$

is *the probability that an* $R_{\mu c}$ *reaction (that is an* R_μ *reaction inside compartment* c*) will occur in the system in* $(t, t+\mathrm{d}t)$*, given that the system is in the state* $(X_{11}, \ldots, X_{NK})$ *at time* t, where

- $r_\mu\,\mathrm{d}t$ is the average probability that a particular molecular pair will react according to R_μ in the infinitesimal time interval $\mathrm{d}t$ inside a compartment whose volume is equal to 1 in the given volume measure unit
- $h_{\mu c}$ is the number of distinct $R_{\mu c}$ molecular reactant combinations available in the state $(X_{11}, \ldots, X_{NK})$ (that is the molecular reactant combinations pertaining reaction R_μ inside compartment c)
- V_c is the volume of compartment c.

According to [5], for a reaction of the kind $S_1 + S_2 \rightarrow S_3$, $h_{\mu c}$ is the product $X_{1c}X_{2c}$, while for a reaction $2S_1 \rightarrow S_2$, $h_{\mu c} = X_{1c}(X_{1c} - 1)/2$.

Finally, by the same considerations stated in [5], the following expression can be easily derived:

$$\mathrm{P}(\tau, (\mu, c)) = \begin{cases} a_{\mu c} \exp(-a_0 \tau) & 0 \leq \tau < +\infty, \\ & \mu = 1, \ldots, M \quad c = 1, \ldots, K \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $a_{\mu c} = V_c^{-1} h_{\mu c} r_\mu$ and $a_0 = \sum_{\nu=1}^{M} \sum_{\gamma=1}^{K} a_{\nu\gamma}$.

3 The Stochastic π@ Language

The MCGM is a transparent generalisation of the original Gillespie model, parameterised w.r.t. the compartments the system is composed of. The price of this generalisation is the insertion, in the implementation of the model, of the informations pertaining the volume size of each compartment, which can be

accomplished in different ways. The most immediate method would be the definition of a function $\mathrm{Vol} : \mathcal{C} \to \mathbb{R}$ returning the volume size of each compartment, but this would not allow to model variable volumes.

The approach chosen here is to allow (but not require) the specification of the volume $\overline{V_i}$ for each element i, which represents *the (average) increment of volume of a compartment c needed for including the additional element i*. In the case of constant volume, $\overline{V_i} = 0$ for each i. In fact, the additional volume required to include any further element is 0 if the volume is constant. Conversely, at constant and homogeneous temperature and pressure, for a single chemical species S occupying volume V, V_i can be calculated as

$$\overline{V_i} = \frac{V}{n} = \frac{m}{n} \cdot \frac{1}{D} = \frac{w_S}{D}$$

where n is the number of molecules of S inside V, m is the total mass, D is the density of S, w_S its molecular weight.

$\overline{V_i}$ is a function depending on the kind of the element i but also on the compartment c, since the chemical composition of c may vary the average distance between the inner elements, because of atomic-level forces like van der Waals interactions and hydrogen bonds. Under the assumption (tolerable if the discussed variation is reasonably small or the chemical composition of the compartments is almost the same) of constant $\overline{V_i}$ for each i, the volume V_c of each compartment c can be easily calculated as the sum of $\overline{V_i}$ for each i inside c, even in the case of exchange of molecules or reorganisations in the compartment structure.

In the next section, the syntax and semantics of Sπ@ is defined according to the above assumptions. In Sect. 4 some useful examples are presented and it is shown how the approach chosen for Sπ@ is both general and simple, since it allows to represent the function $\mathrm{Vol} : \mathcal{C} \to \mathbb{R}$ discussed above as well as the single-compartment situation but also multiple compartments with variable volumes, without increasing the computational complexity of the algorithm, complicating the notation or forcing the introduction of unessential informations in the model.

3.1 Syntax and Semantics

The π@ language [9] is a conservative extension of the π-Calculus [10,11,12] provided with polyadic synchronisation [13] and different levels of priority for actions [14]. The Sπ@ language presented here can be considered in first approximation as the stochastic version of a core π@ limited to two levels of priority and two names for each channel. The capability of giving infinite rates to reactions replaces the two priority levels of this core π@, while the two names denoting each action assume different meaning, since the first represents the type of (chemical) reaction, while the second the compartment where the reaction takes place.

Definition 1. *Let $\mathcal{N}, \mathcal{C}$ be distinct sets of names on finite alphabet, with m, n ranging over $\mathcal{N}$, a, b over $\mathcal{C}$ and x, y over $\mathcal{X} = \mathcal{N} \cup \mathcal{C}$. Let also v range over $\mathbb{R}$ within the interval $[0, +\infty[$. The syntax of the Sπ@ language is defined as*

$$\begin{aligned} P &::= \mathbf{0} \mid \sum_{i \in I} \pi_i.P_i \mid P \mid Q \mid !\pi.P \mid (\nu\, x)P \\ \pi &::= n@a : v(\tilde{x}) \mid \overline{n}@a : v\langle \tilde{x} \rangle \end{aligned}$$

where $\tilde{x}$ represents zero or more names $x_1, \ldots, x_i$ ranging over $\mathcal{X}$.

$\mathbf{0}$ is the null process, capable of doing nothing. $\sum_{i \in I} \pi_i.P_i$, written also $\pi_1.P_1 + \pi_2.P_2$ in the case $|I| = 2$, represents the guarded choice between different actions. $P \mid Q$ means that P and Q are two processes executing in parallel. $!\pi.P$ is the guarded replication. $(\nu\, x)P$ allows the scope restriction of the name x: the restriction of compartment names allows the creation of new compartments, while the restriction of reaction names is used in several ways, like for representing bindings between different elements. The expressions $n@a : v(\tilde{x})$ and $\overline{n}@a : v\langle \tilde{x} \rangle$ represent respectively the polyadic input and output capabilities of a process, where

- n is the kind of reaction the process is ready to perform: in Expr. 2 it corresponds to the index μ denoting the reaction R_μ;
- a is the compartment where the reaction may take place;
- v corresponds to the average volume $\overline{V_i}$ of each element i discussed above and represents the volume occupied inside compartment a by the process ready to perform the input or output action.

For sake of simplicity, the set $\mathcal{N}$ of reaction names is kept distinct from compartment names $\mathcal{C}$. In order to retain syntactical and semantical compatibility with SPiM, the additional informations of compartment and volume are kept optional, that is the grammar is implicitly added with the rules

$$\pi ::= n@a(\tilde{x}) \mid \overline{n}@a\langle \tilde{x} \rangle \mid n : v(\tilde{x}) \mid \overline{n : v}\langle \tilde{x} \rangle \mid n(\tilde{x}) \mid \overline{n}\langle \tilde{x} \rangle$$

which allow to omit volume and compartment informations if not needed. If omitted, the volume v is considered 0, while the compartment c is given a default, distinct value. For sake of simplicity, the additional grammar rules are not considered in the following definitions but their inclusion is straightforward.

Definition 2. *The congruence relation $\equiv$ is defined as the least congruence satisfying alpha conversion, the commutative monoidal laws with respect to both $(\mid, \mathbf{0})$ and $(+, \mathbf{0})$ and the following axioms:*

$$\begin{aligned} (\nu\, x)P \mid Q &\equiv (\nu\, x)(P \mid Q) && \text{if } x \notin fn(Q) \\ (\nu\, x)P &\equiv P && \text{if } x \notin fn(P) \\ !\pi.P &\equiv \pi.(\, !\pi.P \mid P) \end{aligned}$$

where the function fn *is defined as*

$$\mathrm{fn}(n@a\!:\!v(\tilde{x})) \stackrel{def}{=} \{n,a\} \qquad \mathrm{fn}(\overline{n}@a\!:\!v\langle\tilde{x}\rangle) \stackrel{def}{=} \{n,a,\tilde{x}\}$$

$$\mathrm{fn}(\mathbf{0}) \stackrel{def}{=} \emptyset \qquad \mathrm{fn}((\nu\ x)P) \stackrel{def}{=} \mathrm{fn}(P)\setminus\{x\}$$

$$\mathrm{fn}(\pi.P) \stackrel{def}{=} \mathrm{fn}(\pi)\cup\mathrm{fn}(P) \qquad \mathrm{fn}(\sum_{i\in I}\pi_i.P_i) \stackrel{def}{=} \bigcup_i \mathrm{fn}(\pi_i.P_i)$$

$$\mathrm{fn}(P \mid Q) \stackrel{def}{=} \mathrm{fn}(P)\cup\mathrm{fn}(Q) \qquad \mathrm{fn}(!\pi.P) \stackrel{def}{=} \mathrm{fn}(\pi.P)$$

Definition 3. *S*π*@ semantics is given in terms of the following reduction system:*

$$(C)\ \frac{}{(n@a\!:\!v_1(\tilde{x}).P+M) \mid (\overline{n}@a\!:\!v_2\langle\tilde{y}\rangle.Q+N) \xrightarrow{\mathrm{rate}(n)} P\{\tilde{y}/\tilde{x}\} \mid Q}$$

$$(R)\ \frac{P \xrightarrow{r} P'}{(\nu\ x)P \xrightarrow{r} (\nu\ x)P'} \qquad (P)\ \frac{P \xrightarrow{r} P'}{P \mid Q \xrightarrow{r} P' \mid Q} \qquad (E)\ \frac{P\equiv Q \quad P \xrightarrow{r} P' \quad P'\equiv Q'}{Q \xrightarrow{r} Q'}$$

The rule (C) allows the communication of the names $\tilde{x}$ from process P to Q, where they are properly substituted to names $\tilde{y}$. The function rate : $\mathcal{N} \to (\mathbb{R}\cup+\infty)$ is an external function which permits to associate the correct rate to each reaction, where the rate corresponds to the value r_μ of Expr. 2. Rules $(R),(P),(E)$ allow the transition of processes in presence of restriction, parallel operator or by exploiting structural equivalence.

Definition 4. *A* Sπ*@ system* S *is said to be in* standard form *if*

$$S = (\nu\ \tilde{x})\big(P_1 \mid \cdots \mid P_j \mid !P_{j+1} \mid \cdots \mid !P_k\big)$$

and each P_i *is a non-empty sum.*

Proposition 1. *For every* Sπ*@ system* S*, there exists a system* S' *such that* $S\equiv S'$ *and* S' *is in standard form.*

In order to calculate the value $h_{\mu c}$ of Expr. 2, we introduce, according to [4], the function Act which permits to know the number of possible combinations of inputs and outputs on a reaction channel inside a given compartment.

Definition 5. *The activity* Act *of channel* n *inside compartment* a *in the system* S *is defined as*

$$\mathrm{Act}_{n@a}(S) = (\mathrm{In}_{n@a}(S)\cdot\mathrm{Out}_{n@a}(S)) - \mathrm{Mix}_{n@a}(S)$$

where S *is in standard form,* $\mathrm{In}_{n@a}(S)$ *and* $\mathrm{Out}_{n@a}(S)$ *are the number of unguarded inputs and outputs on channel* n *inside compartment* a*, and* $\mathrm{Mix}_{n@a}(S)$ *is the sum of* $\mathrm{In}_{n@a}(\sum_i)\cdot\mathrm{Out}_{n@a}(\sum_i)$ *for each summation* $\sum_i$ *in* S*.*

The function chan allows to know all the active channels inside each compartment in a given system S.

Definition 6. *Given a Sπ@ system S in standard form*

$$S = (\nu\ \tilde{x})\big(P_1 \mid \cdots \mid P_j \mid\ !P_{j+1} \mid \cdots \mid\ !P_k\big)$$

the function chan *is defined recursively as follows:*

$$\mathrm{chan}(S) = \bigcup_{i=1}^{k} \mathrm{chan}(P_i) \qquad \mathrm{chan}(\sum_{i\in I} \pi_i.P_i) = \bigcup_{i\in I} \mathrm{chan}(\pi_i)$$
$$\mathrm{chan}(n@a : v(\tilde{x})) = \mathrm{chan}(\overline{n}@a : v\langle\tilde{x}\rangle) = \{n@a\}$$

The volume V_c of Expr. 2 is calculated as the sum of the volumes occupied by each process ready to perform a reaction inside compartment c. It is worth noticing that in the case of nested compartments as in Fig. 1 (a), the volumes of the compartments inside c must not be summated to V_c in order to respect Expr. 2, since the elements of c are not allowed to diffuse into the inner compartments and consequently their concentration does not depend on inner compartments volumes.

Definition 7. *Given a Sπ@ system S in standard form*

$$S = (\nu\ \tilde{x})\big(P_1 \mid \cdots \mid P_j \mid\ !P_{j+1} \mid \cdots \mid\ !P_k\big)$$

the volume Vol_a *of the compartment a in the system S is calculated as follows:*

$$\mathrm{Vol}_a(S) = \sum_{i=1}^{k} \mathrm{Vol}_a(P_i)$$
$$\mathrm{Vol}_a(\sum_{i\in I} \pi_i.P_i) = \sum_{i\in I} \mathrm{Vol}_a(\pi_i)$$
$$\mathrm{Vol}_a(\overline{n}@a : v\langle\tilde{x}\rangle) = \mathrm{Vol}_a(n@a : v(\tilde{x}))$$
$$\mathrm{Vol}_a(n@b : v(\tilde{x})) = \begin{cases} v & a = b \\ 0 & \text{otherwise} \end{cases}$$

If $\mathrm{Vol}_a(S) = 0$*, then a is given the default volume value* 1.

When $\mathrm{Vol}_a(S) = 0$, all the informations on the volumes v occupied by the reactants inside a have been omitted. This is the case of modelling in Spi, hence it is quite natural to redefine $\mathrm{Vol}_a(S)$ as 1 so that the reaction rates defined for Spi programs coincide with the reaction rates calculated in Sπ@.

The volume of a compartment depends on *all* the enclosed objects, not only on reacting ones. Consequently, the volume of non reacting elements must be taken in account and inserted into the program when such volume informations are relevant for the simulation.

Finally, the multi-compartment stochastic simulation algorithm (MSSA) is defined as extension of the SSA given in [5], according to the previous definitions and the MCGM:

Definition 8. *Given a Sπ@ system S in standard form, the selection of the next reaction* $\mathrm{Next}(S)$ *and of the delay* $\mathrm{Delay}(S)$ *relative to the MSSA are described by the following algorithm:*

1. *For each channel* c_i *in* $\mathrm{chan}(S)$*, with* $\mathrm{chan}(S) = \{c_1, \ldots, c_j\}$*, calculate*

$$a_i = \mathrm{Act}_{n@b}(S) * \mathrm{rate}(n) / \mathrm{Vol}_b(S)$$

   where $c_i = n@b$ *for some* $n \in \mathcal{N}, b \in \mathcal{C}$*.*
2. *Calculate* $a_0 = \sum_{i=1}^{j} a_i$
3. *Generate two random numbers* $k_1, k_2 \in [0, 1]$ *and calculate* τ, μ *such that*

$$\tau = (1/a_0) \ln(1/k_1) \qquad \sum_{i=1}^{\mu-1} a_i < k_2 a_0 \leq \sum_{i=1}^{\mu} a_i$$

4. $\mathrm{Next}(S) = c_\mu$ *and* $\mathrm{Delay}(S) = \tau$*.*

The value $c_\mu = n@b$ for some n, b denotes the reaction channel n and the compartment c of the next reaction happening after τ time. The two processes performing the synchronisation step on c_μ are then randomly chosen as for SPiM.

As noted in [4] for the activities Act, the volumes Vol can be calculated by difference with the respective volumes of the previous execution step, so that the computational complexity order of the algorithm is unvaried.

4 Examples

In this section some examples of chemical and biological modelling are presented, in order to illustrate the simplicity of the language and the increased biochemical faithfulness.

4.1 Chemical Reactions

If m_1 and m_2 are two molecules floating inside the only, fixed-size compartment of the system, it is quite useless to specify the compartment they are in and the volume they occupy: this is the exact case of modelling in Spi. As discussed after Def. 1, the system composed of m_1 and m_2 could easily be written in Sπ@ as well:

$$M_1 \equiv r_1.N \qquad M_2 \equiv \overline{r}_1 \qquad S \equiv M_1 \mid M_2$$

where m_1 and m_2 are supposed to undergo a reaction of type R_1 (represented by channel r_1) and produce a new chemical reactant n. As previously discussed, in absence of volume and compartment informations, the volume of the reactants is considered as 0, while that of the (only, unspecified) compartment is considered as 1, so that also the function $\mathrm{rate}(r_1)$ is consistent – and actually coincides, according to Expr. 2 – with the rate given for the simulation in Spi.

The situation in Fig. 1 (a), which presents multiple compartments could be encoded as

$$M_{11} \equiv r_1@c_1.N(c_1) \quad M_{21} \equiv \overline{r}_1@c_1 \qquad M_{12} \equiv r_1@c_2.N(c_2) \quad M_{22} \equiv \overline{r}_1@c_2$$
$$S \equiv M_{11} \mid M_{21} \mid M_{12} \mid M_{22}$$

if supposing c_1 and c_2 of the same, fixed-size. If their volumes v_1, v_2 are constant, but different, it is possible to specify them by adding two further elements d_1, d_2 which do not take part in the reactions but allow to insert the desired informations:

$$M_{11} \equiv r_1@c_1.N(c_1) \quad M_{21} \equiv \overline{r}_1@c_1 \qquad M_{12} \equiv r_1@c_2.N(c_2) \quad M_{22} \equiv \overline{r}_1@c_2$$
$$D_1 \equiv (\nu\, n)n@c_1 : v_1 \qquad D_2 \equiv (\nu\, n)n@c_2 : v_2$$
$$S \equiv M_{11} \mid M_{21} \mid M_{12} \mid M_{22} \mid D_1 \mid D_2$$

Even if the processes D_1, D_2 are unable to perform any reaction, v_1 and v_2 are taken into account when calculating c_1 and c_2 volumes. According to Def. 7, since the volume occupied by each of the other processes is 0, c_1 and c_2 volumes turn out to be v_1 and v_2 respectively, as required. In this way the volumes of K compartments (and consequently the rate of each reaction inside them) can be specified directly inside the program by adding not more than K non-reactive elements. Alternatively, it is possible to specify the volumes V_{m_1}, V_{m_2} occupied by each molecule and ensure that the volume of their chemical product n is equal to the sum of theirs (as it approximately happens in many real reactions). In general, it is possible to specify only the volumes of the desired elements while keeping consistent compartment volumes with reaction rates.

One of the major advantages of Sπ@ approach is the possibility to specify once the behaviour of one element and create instances of it inside any compartment, which is very convenient when the same objects are present in several different places. The previous system could be rewritten as

$$T_1 \equiv !m_1(c).r_1@c.N(c) \qquad T_2 \equiv !m_2(c).\overline{r_1}@c$$
$$S \equiv \overline{m_1}\langle c_1\rangle \mid \overline{m_1}\langle c_2\rangle \mid \overline{m_1}\langle c_1\rangle \mid \overline{m_1}\langle c_2\rangle \mid T_1 \mid T_2 \mid D_1 \mid D_2$$

by exploiting the possibility of giving infinite rate to channels m_1, m_2 so that the creation of the instances is immediate.

4.2 Biological Modellings

In this section we outline the way some simple biological entities and phenomena [15] can be intuitively represented in Sπ@. The interaction or exchanging of molecules between compartments constitute the easier and more interesting situations.

Membrane Receptor. Consider the simple scheme of a cell membrane receptor whose behaviour is to detect molecules of kind m_1 outside the cell and signal

their presence inside the cell by transforming molecules of kind m_2 into m_3, as in Fig. 1 (b). If we use the membrane as compartment boundary, by modelling the space outside the cell as compartment e and c as the cell itself, the receptor – which is a transmembrane protein – lies partially in both compartments. This means that the process R representing the receptor needs to know the names of the two compartments and to use them for interacting both outside and inside the cell. The system could be for example encoded as

$$M_1 \equiv \overline{bind\_m_1}@e \quad M_2 \equiv conv\_m_2@c \quad R \equiv !bind\_m_1@e.\overline{conv\_m_2}@c.M_3(c)$$

where the names $bind\_m_1, conv\_m_2$ carry the interaction between the receptor R and the molecules m_1 and m_2. The process $M_3(c)$ represents the transformed molecule m_3 inside compartment c.

Pump. The conveyance of molecules or ions across a membrane constitutes a simple example for showing how the exchange of elements between compartments is straightforward in Sπ@: the sodium/potassium pump represents a quite classical modelling candidate. The Na^+/K^+ ATPase (Fig 1 (c)) is a ion pump which moves three Na^+ ions out and two K^+ ions into the cell for each consumed ATP molecule. The pump can be schematised by process P

$$\begin{aligned} ATP(d) &\equiv \overline{atp}@d & P \equiv\ & !atp@c.na@c.na@c.na@c. \\ ADP(d) &\equiv \overline{adp}@d & & \big(ADP(c) \mid k@e.k@e. \\ NA(d) &\equiv \overline{na}@d & & \quad (NA(e) \mid NA(e) \mid NA(e) \mid \\ K(d) &\equiv \overline{k}@d & & \quad K(c) \mid K(c)\big) \end{aligned}$$

which first recruit the ATP molecule, then binds the three sodium ions inside the cell, releases the ADP molecule deriving from the hydrolysed ATP, binds the two K^+ ions outside the cell and finally releases the captured ions into the proper compartments.

Osmosis. The last considered example allows to observe the limits of the previous models and the need to introduce volume information specified in Expr. 2, in order to manage correctly the relative rates of reactions in function not only of the number of elements for each reactant, but also of its concentration. The system in Fig. 1 (d) depicts a compartment c of variable size containing a water solution placed in a hypotonic environment e. The compartment is bounded by a semipermeable membrane, i.e. a filter which allows only the water to move across. Experience shows that this situation causes a net movement of water towards compartment c, due to the so called osmosis phenomenon. Biological occurrence of this circumstance is common both in animal and vegetal cells. A typical animal cell swells when placed in hypotonic and shrinks when in hypertonic solution even if the cell membrane is poorly permeable to water, thanks to the presence of water-channel proteins which allow only water to flow, in either

directions. The process A may describe a simple abstraction of water-channel protein acquaporin:

$$HOH(d) \equiv \overline{hoh}@d : v_{H_2O} \qquad S(d) \equiv \overline{s}@d : v_S$$
$$A(f,g) \equiv !hoh@f.HOH(g) \mid !hoh@g.HOH(f)$$

S is the solute, HOH represents a water molecule able to interact with the acquaporin A and move through the membrane of the cell. In this very simple layout, the acquaporin is completely symmetrical. The system in Fig. 1 (d) may be written as

$$Sys \equiv \underbrace{HOH(e) \mid \cdots \mid HOH(e)}_{m_1} \mid \underbrace{S(e) \mid \cdots \mid S(e)}_{n_1} \mid A(c,e) \mid$$
$$\underbrace{HOH(c) \mid \cdots \mid HOH(c)}_{m_2} \mid \underbrace{S(c) \mid \cdots \mid S(c)}_{n_2}$$

where m_1 and n_1 represent the number of water molecules and salt ions outside, m_2 and n_2 the number of molecules and ions inside the cell membrane and only one acquaporin is present, for simplicity. If we discard the informations on the volumes V_e and V_c of the two compartments in Expr. 2, we are first forced to introduce asymmetry in the encoding of the acquaporin A, in order to differentiate the rate r_e of molecules entering from the rate r_c of molecules leaving the cell. The system can be considered in equilibrium when the the probability of a water molecule entering is equal to the probability of a water molecule leaving the cell, that is when

$$h'_e r_e = h'_c r_c \tag{4}$$

where h'_e is the number of possible combinations of acquaporin–water molecule outside, h'_c the possible combinations inside the cell. Since only one acquaporin is present, we have that $h_e = m'_1$ and $h_c = m'_2$, where m'_1 and m'_2 are the number of water molecules at equilibrium. Hence Expr. 4 becomes

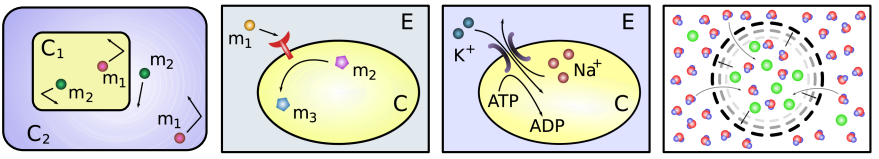
$$m'_2/m'_1 = r_e/r_c$$

meaning that the equilibrium depends on the rate of the molecules initially conveyed, which is not true. In reality, under the hypothesis of uniform temperature and pressure, the equilibrium is reached when the concentration of the two solutions is the same, that is when

$$m'_2/n'_2 = m'_1/n'_1 \tag{5}$$

where n'_1 and n'_2 are the number of salt ions in the respective compartments. Conversely, this dependence is coherently modeled if we consider the right expressions for V_e and V_c. In fact, by Expr. 2, we have

$$V_e^{-1} h'_e r_e = V_c^{-1} h'_c r_c \tag{6}$$



(a) (b) (c) (d)

Fig. 1. (a) Two equivalent molecule pairs floating in compartments of different volume. (b) Membrane receptor. (c) Ion Pump. (d) Osmosis by semipermeable membrane in hypotonic solution.

Since the volumes are obtained as the sum of the average volume occupied by each element, we have

$$V_e = \sum_e v_{H_2O} + \sum_e v_S = m'_1 v_{H_2O} + n'_1 v_S$$
$$V_c = \sum_c v_{H_2O} + \sum_c v_S = m'_2 v_{H_2O} + n'_2 v_S$$

which – under the initial symmetric assumption $r_e = r_c$ – properly substituted in 6 leads to

$$\frac{1}{v_{H2_O} + \frac{n'_1}{m'_1} v_S} = \frac{1}{v_{H2_O} + \frac{n'_2}{m'_2} v_S}$$

that is clearly satisfied by equation 5.

Osmosis involves most of living cells, and expression (5) describes only one of the possible equilibrium conditions. For example, plant cells are surrounded by rigid walls which prevent them from increasing their volume. Consequently, if placed in hypotonic solution, these cells absorb water until the pressure on cell walls equals the osmotic pressure, which depends on the absolute temperature T and the difference of salt ions/molecules concentration. This equilibrium condition cannot be expressed in Sπ@, since any pressure evaluation lacks. Such kind of informations may be introduced in the model by defining reaction rates as functions of the absolute temperature T and pressure p_c of the compartment, where p_c may be in turn calculated as function of the compartment c, the absolute temperature and the elements surrounded by c. This would allow to take into account temperature, pressure and some of the structural informations pertaining mechanical properties of compartment boundaries. Depending on the kind of function used for evaluating p_c, the computational complexity of the algorithm may grow significantly.

5 Related Work

Extensions of the SSA handling varying volumes were already considered in [16,17]. The chosen approach consists in the expression of the propensity functions as known,

time-dependent functions. While this approach provides faster simulations, it can be applied only in the case that the variation is deterministic and it is known in advance. Conversely, the MSSA allows to introduce the variation of volumes in the *discrete* and *stochastic* behaviour of the system, in perfect agreement with the intention of the original SSA.

The next subvolume method (NSM) [18] handles multiple volumes, in order to provide efficient simulation of chemical systems in presence of molecular diffusion. The NSM is thought for a high number of volumes with statical structure, of fixed and equal size. Although the NSM is faster than the MSSA, it does not handle dynamical compartments with varying volumes and cannot be directly extended with this aim without losing its computational efficiency.

6 Conclusion

The osmosis example shows a simple (and biologically common) situation where the SSA happens not to be faithful if taken "as it is", since the volume of each element (and not only compartment volumes) must be properly considered during the simulation in order to obtain the correct values for reaction rates in function of reactants concentration. The reason is that SSA already embodies the unique, fixed-size volume hypothesis, which is correct for fluid systems under the conditions stated in [5] and needs to be properly translated into another chemical or biological context. The presented adaptation of the SSA to multiple compartments allows to take in account the informations on the concentration of reactants in function of volumes without changing the underlying probabilistic/kinetic model, so that the MSSA results somehow closer to the biological scenario. Anyway, Sπ@ modelling is still far from being complete w.r.t. biological reality, for a number of reasons. Here are some:

- temperature and pressure are considered constant and uniform, reasonable condition in most (but not all) of the biological situations, as discussed in the next section;
- many atomic-level phenomena (e.g. van der Waals or electrostatic effects, hydrogen bonds, ...) are disregarded in the kinetic model;
- all the particles are considered to be small, free floating and homogeneously distributed in their respective compartments while many biological elements are not small, neither homogeneously distributed nor allowed (or only partially) to move freely;
- all the physical and dimensional informations on biological elements and structures (e.g. properties like tridimensional position or shape, or surface area, fluidity, thickness of membranes, mechanical resistance, elasticity, ...) cannot be modeled.

In addition, Sπ@ inherits two significant limits – strict seriality and not more than two reactants (but unlimited products) for each reaction, which are due to the point-to-point, interleaving synchronisation semantics of the π-Calculus.

On the other hand the presented approach has several advantages. The most important is the capability of modelling dynamical compartment structures. This

can be achieved by preserving several compartment semantics, like for example atonality of BioAmbients [3] or bitonality of Brane Calculi [6]. In fact, both these calculi can be reproduced by adapting the encodings presented in [9] to Sπ@.

Despite of the increased expressiveness, the calculus is very conservative. The original kinetic/stochastic hypotheses of Gillespie's model are completely preserved and Sπ@ semantics is very close to that of the stochastic π-Calculus. This allows to obtain a correct Sπ@ machine by simple extension of SPiM and also to easily adapt the related graphical representation [19].

Beyond the fact that SPiM [4] programs can be directly executed on Sπ@ by retaining the same semantics, the additional informations pertaining compartment and volume can be inserted only when necessary and only for the desired elements and do not increase the computational complexity of the original Gillespie algorithm. Fixed-size or variable volume, single or multiple compartments can be modeled but reaction rates have a unique representation, independent of (but coherent with the volume of) the compartment where the reaction happens. In the same way elements can be defined once and easily instantiated in any compartment, or moved across without readjusting their behaviour, typical feature only of calculi with explicit compartment semantics like BioAmbients or Brane.

The inclusion of further physical properties (like temperature and pressure) and the parallelisation of the algorithm in the way of [20] by Gillespie's tau leaping [21] are left for future work.

References

1. Regev, A., Silverman, W., Shapiro, E.Y.: Representation and simulation of biochemical processes using the pi-calculus process algebra. In: Pacific Symposium on Biocomputing, pp. 459–470 (2001)
2. Priami, C., Regev, A., Shapiro, E.Y., Silverman, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Inf. Process. Lett. 80(1), 25–31 (2001)
3. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.Y.: Bioambients: an abstraction for biological compartments. Theor. Comput. Sci. 325(1), 141–167 (2004)
4. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. In: Bioconcur'04, ENTCS (2004)
5. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81(25), 2340–2361 (1977)
6. Cardelli, L.: Brane calculi. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 257–278. Springer, Heidelberg (2005) [22]
7. Priami, C., Quaglia, P.: Beta binders for biological interactions. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 20–33. Springer, Heidelberg (2005) [22]
8. Cazzaniga, P., Pescini, D., Romero-Campero, F.J., Besozzi, D., Mauri, G.: Stochastic approaches in P systems for simulating biological systems. In: Gutiérrez-Naranjo, M.A., Paun, G., Riscos-Núñez, A., Romero-Campero, F.J. (eds.) Fourth Brainstorming Week on Membrane Computing, Sevilla, Fénix Editora, January 30 - February 3, 2006, vol. I, pp. 145–164 (2006)

9. Versari, C.: A core calculus for a comparative analysis of bio-inspired calculi (2007), http://www.cs.unibo.it/~versari/files/cversari-esop07.pdf
10. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, i. Inf. Comput. 100(1), 1–40 (1992)
11. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, ii. Inf. Comput. 100(1), 41–77 (1992)
12. Milner, R.: Communicating and mobile systems: the π-calculus. Cambridge University Press, New York, NY, USA (1999)
13. Carbone, M., Maffeis, S.: On the expressive power of polyadic synchronisation in pi-calculus. Nord. J. Comput. 10(2), 70–98 (2003)
14. Cleaveland, R., Lüttgen, G., Natarajan, V.: Priority in process algebra. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebra, pp. 711–765. Elsevier Science Publishers, Amsterdam (2001)
15. Lodish, H., Berk, A., Matsudaira, P., Kaiser, C.A., Krieger, M., Scott, M.P., Zipursky, L., Darnell, J.: Molecular Cell Biology. W. H. Freeman, New York (2004)
16. Lu, T., Volfson, D., Tsimring, L., Hasty, J.: Cellular growth and division in the gillespie algorithm. In: Systems Biology, IEE Proceedings, pp. 121–128 (2004)
17. Lecca, P.: A time-dependent extension of gillespie algorithm for biochemical stochastic π-calculus. In: SAC '06: Proceedings of the 2006 ACM symposium on Applied computing, New York, NY, USA, pp. 137–144. ACM Press, New York (2006)
18. Elf, J., Ehrenberg, M.: Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. Systems Biology 1(2), 230–236 (2004)
19. Phillips, A., Cardelli, L., Castagna, G.: A graphical representation for biological processes in the stochastic pi-calculus, vol. 4230, pp. 123–152 (2006)
20. Cazzaniga, P., Pescini, D., Besozzi, D., Mauri, G.: Tau leaping stochastic simulation method in p systems. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2006. LNCS, vol. 4361, pp. 298–313. Springer, Heidelberg (2006)
21. Cao, Y., Gillespie, D.T., Petzold, L.R.: Efficient step size selection for the tau-leaping simulation method. Journal of Chemical Physics 124(4) (2006)
22. Danos, V., Schachter, V. (eds.): CMSB 2004. LNCS (LNBI), vol. 3082, pp. 26–28. Springer, Heidelberg (2005)

Computational Simulation of Optical Tracking of Cell Populations Using Quantum Dot Fluorophores

Martyn R. Brown[1], Paul Rees[1], Steve Wilks[1], Huw D. Summers[2], Rachel J. Errington[3], Kerenza L. Njoh[3], Sally C. Chappell[3], Paul J. Smith[3], and James F. Leary[4]

[1] Multidisciplinary Nanotechnology Centre, Swansea University, Singleton Park, Swansea, SA2 8PP, U.K.
m.r.brown@swansea.ac.uk
[2] School of Physics and Astronomy, Cardiff University, 5, The Parade, Cardiff, CF24 3YB, U.K.
[3] School of Medicine, Cardiff University, Heath Park, Cardiff, CF14 4XN, U.K.
[4] Birck Nanotechnology Centre, Purdue University, West Lafayette, Indiana, IN 47907, U.S.

Abstract. Quantum dot fluorophores provide a photo and bio-stable optical marker signal well suited to the tracking of lineage within large cell populations over multiple generations. We have used a Monte Carlo algorithm to model the process of dot partitioning and dilution by cell mitosis. A Genetic Algorithm was used to compare simulated and experiment quantum dot distributions, which shows that the dot fluorescence is divided with a stochastic variation about an asymmetric mean split ratio.

Keywords: Cell division, Quantum dot fluorophores, Monte-Carlo simulation, Genetic algorithm.

1 Introduction

The ability to track the evolution of large cell populations over time is crucial, not only for providing a means of monitoring the general health of a population of cells, but also for informing on the outcome of specific assays (e.g. pharmacodynamic assay). The overall aim of the current study has been to determine if a targeted quantum dot (QD) fluorescent reporter system provides labeling of cell proliferation independent of cell cycle. A key component to this has been the computational simulation of the QD dilution via cell mitosis. Modeling of this kind provides detailed insights into the evolution of cell lineage; providing insight at the individual cell level from whole population experiments i.e. flow cytometry analysis as opposed to cell to cell tracking via time-lapse imaging. Traditional approaches used for determining cell proliferation require knowledge of population size or the behavior of a cellular marker diluted on a cell-to-cell basis. The latter approach can utilize samples of populations and a frequently used chemical marker is based on an organic fluorophore CFSE (carboxy-fluorescein diacetate succinimidyl ester). CSFE diffuses freely into cells where intracellular esterases cleave the acetate groups converting it to a fluorescent,

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 96–105, 2007.

membrane-impermeant dye that is retained within the cytoplasm of the cell (1,2). During each round of subsequent cell division, the relative intensity of the dye fluorescence is halved. Cells can be analyzed by flow cytometry to determine the intensity distribution of the fluorophore signal within the cell population and thereby quantify the extent of proliferation (3). As with all organic fluorophores CFSE has inherent disadvantages of bio and photo-instability when used in longer-term live cell assays. We have therefore considered the use of QDs (inorganic nanocrystals). The advantages of using QDs in the place of traditional organic fluorophores have been widely reported (4). Firstly, they are photostable (5), allowing long-term labeling of live cell populations. Secondly QDs have a broad-band absorption meaning that when using multi-color labeling cells a single excitation wavelength can be used (6). They have a major advantage over conventional organic fluorophores: they are chemically stable and are not metabolized by the cell. Not only is their fluorescent signal very much brighter than an organic fluorophore (7,8) but it should also remain unperturbed by intra-cellular bio-chemical reactions. It is the stability and persistence of the QD signal within cells over multiple generations that we are exploiting. The objective is to develop a robust assay of integrated fluorescence cellular readout which reports the extent of cellular bifurcation within a complex population (ie cell division and lineages), potentially providing profiles of drug resistance, cell clonality and levels of aneuploidy in complex tumour populations. In particular we have exploited the tracking capability to detect proliferative sub-fractions within tumour populations (human osteosarcoma cell line U-2 OS; ATCC HTB-96) subjected to a cell-cycle perturbing drug ICRF-193, that has the capacity to inhibit cell division. We have used the commercially available Qtracker® 705 system which delivers CdTe/ZnS core-shell QDs, with peak fluorescent emission at 705 nm, to the endosomal compartment and provides long term tracking potential (> six sequential generations).

2 Experimental Results

Time-lapse microscopy experiments were carried out to determine whether the endosomal targeted QDs had any perturbing effect on the U-2 OS population. By conducting single cell analysis rather than whole population cell counting we were able to detect small and transient perturbations of cell growth and relate this to cell cycle induced stress responses (9). A mitotic event curve was derived from measuring cumulative time to mitosis of individual cells, each contributing to a kinetic picture of the population response. U-2 OS ATCC cells were loaded with a single concentration of Qtracker® 705 (4 nM), cell growth was determined via transmission time-lapse microscopy for a period of 24 hours (10). Manual image analysis provided the ability to extract the time-to-event curves which demonstrated the dynamics of event delivery in quantum dot loaded and control conditions. The process of targeted labeling with QDs showed no acute effects on the ability of the cells to traverse the cell cycle and deliver to mitosis. The carrier alone showed a slight perturbation (between 7 to 14 hours) with a subsequent recovery over the remaining 10 hours. In control conditions all of the events during the course of the sequence were successful cell divisions leading to two daughters. We conclude that there is a minimal

perturbation to the cell cycle when loading near-infrared quantum dots via the endosomal pathway.

Studies of large cell samples (10^4 cells) were carried out using flow cytometry. Tracking of the population via the QD fluorescence was carried out at 24 hr intervals after an initial 24 hr period to allow for stabilization of the signal following dot uptake. A typical set of histograms is shown in figure 1a, these data have been gated using the side and forward cell scatter signals to identify the viable cell component and thus remove debris from the sample set (practically this entails removing measurements at the low and high end limits of the scattered light intensity range). The endosomal targeting of the QDs produces a large dynamic range in the dot fluorescence and hence the acquisition using a logarithmic detection scale.

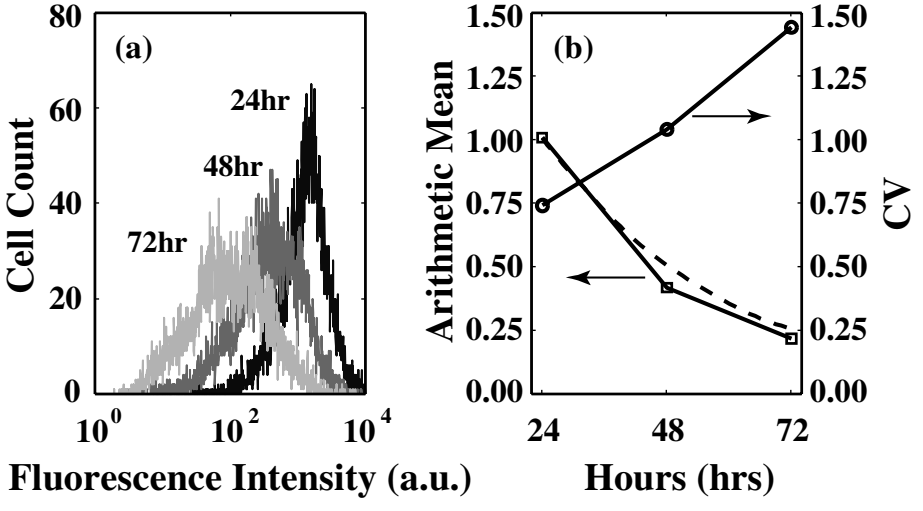


Fig. 1. a.) Quantum dot fluorescence intensity histograms taken at 24 hour intervals following take-up; b.) Statistics from the histograms: arithmetic mean (solid line with squares) and coefficient of variation (solid line with circle). The dotted line represents the expected mean due to halving of the dot density per cell every 24 hours.

As the assay progresses the histograms move down on the x-axis and broaden leading to a reduced peak cell count. The loss of fluorescence signal is expected as the dot density per cell becomes diluted by partitioning upon cell division. The mean inter-mitotic time (IMT), calculated from time lapse images' is 22.5 hours with a standard deviation of 6 hous. If the dot partitioning is symmetric, holding to a ~ 50:50 split in the percentage of dots passed onto daughter cells, then the distribution should shift to half the x-axis value with an unchanged peak cell count (all histograms have the same sample number of 10^4). The broadening of the distribution therefore indicates that dot partitioning is asymmetric and the flow data supports the observation of asymmetric splitting in the microscope images. At this point in our studies it is not clear whether the asymmetry is 'biological' relating to uneven partitioning of organelles or 'extrinsic' and due to uneven take up of dots within

endosomes. The physical mechanism of the asymmetry is unimportant in relation to the cell proliferation studies reported in this paper as the validity of the QD assay is based purely on tracking of their fluorescence. To quantify the distributions we use two statistical measures; the arithmetic mean ($\bar{x}$) which can be interpreted as the average dot fluorescence per cell and the co-efficient of variation ($CV = \sigma/\bar{x}$) which is a measure of the distribution width relative to its mean value. The values of $\bar{x}$, normalized to the 24 hour value, and the CV are plotted in figure 1b. The mean should halve in each 24 hour period as QDs are diluted between daughter cell pairs; the solid line in the figure shows an exact halving and the measured fluorescence signal is within ~ 10% of this thus confirming that the dots do provide a persistent and stable optical tracking signal. The CV doubles between 24 and 72 hours due to the asymmetric division of the QDs (for symmetric partitioning $\bar{x}$ and σ halve upon division). These simple, statistical descriptors of the fluorescence histograms are therefore powerful analytical outputs, validating the integrity of the tracking signal and providing an insight to the mechanics of the dot dilution by the growing cell population.

3 Computational Simulations

3.1 Overview

A more sophisticated data analysis has been performed based on a stochastic model of cell mitosis within a population in order to simulate the fluorescence histogram evolution and so provide a more detailed picture of the quantum dot partitioning during cell division. The QD fluorescence data taken at 24 hours was used as an input set for this computational analysis. From this data we determine the number of dots allocated to each of the 10^4 cells (the number of dots is taken to be proportional to the fluorescent intensity). The model assumes that these cells are randomly distributed uniformly through the cell cycle period. The model increments time using a time step determined in order to correctly predict the statistics of cell mitosis. A Monte Carlo algorithm is then applied to determine whether or not a cell splits during this time increment. The algorithm utilizes a Gaussian distributed inter-mitotic time distribution (mean IMT = 22.5 hours, σ = 6 hours) measured using lapse measurements referred to above. A normal distribution was used for two reasons: (i) we are evaluating a large data set, i.e. > 10^4 cells and thus is a good approximation to the more commonly used binomial distribution and, (ii) due to the very nature of the cell splitting process there is an intrinsic variance of the mean splitting ratio between cells and a normal distribution allows this to be easily adopted with in the simulation.

On splitting we assume that the number of quantum dots is conserved i.e. the total number of dots in each daughter cell is equal to the number of dots in the parent cell. The number of dots allocated to each daughter cell is chosen at random from a Gaussian distribution of division ratios. At the set ‘measurement’ time a fluorescent histogram is calculated by determining the number of dots in each cell, this histogram can then be compared directly with the experimental data.

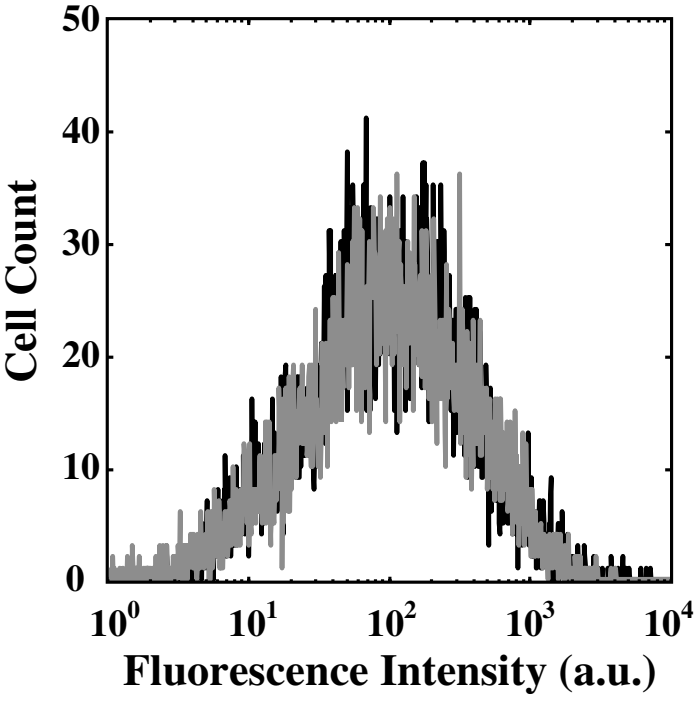


Fig. 2. Computed (grey trace) and measured (black trace) fluorescence histograms 72 hours after quantum dot uptake. The modeled fit has a peak probability of partitioning ratio of 74:26 % with a 6 % standard deviation.

The correct choice of Gaussian distribution of division ratios i.e. the mean and standard deviation is determined by using a genetic algorithm (11) to best fit the data. The fitness of the modeled histograms was determined by least squares fitting of calculated and measured data sets. The modeled histogram at 72 hours, computed from the 24 hour data, is shown in figure 2 together with the measured distribution. This fit is achieved with an asymmetric fluorescence partitioning of 74:26 % with a standard deviation of 6 %. The two data sets are virtually indistinguishable and so this substantiates the experimental observation that the dot dilution is an asymmetric process and indicates that there is a stochastic variation about the mean.

3.2 Details of the Simulation

A flow chart depicting the main steps of the stochastic cell-splitting algorithm is shown in figure 3. Firstly, the recorded data describing the fluorescence of the quantum dots within the cells at 24 hours is taken as an input set for the program. Each of the 10^4 input cells is stochastically given a cycle time, which is determined randomly from a normal distribution centered on the IMT and with a standard deviation of 6 hours. This step mimics the fact that each of the 10^4 cells in the experiment will be at a different stage of the cell cycle. The cycle time of each cell is

incremented in 1 hour time steps in conjunction with the laboratory frame of each time a loop of the algorithm is completed reference.

The next step of the algorithm determines if a particular parent cell has split or not. Again this choice is stochastically determined, where the previously assigned cycle time of a cell together with the laboratory time is used to generate a cumulative distribution specific to the individual cell in question. In figure 4 below, a case where a particular cell has been given a cycle time of 12.5 hours, resulting in a cumulative distribution centered on 35 hours, this point and the standard deviation range (dot-dash line and dashed lines respectively) are indicated on the figure.

A splitting event occurs if a random number, uniformly distributed over the interval [0 1], lies below the cumulative frequency curve at the laboratory time. For example, the filled black circle indicates the probability of a split occurring for this particular cell at a laboratory time of 27 hours, the graph indicates a 10% chance of this split occurring. This stochastic sampling is applied to each cell every hour.

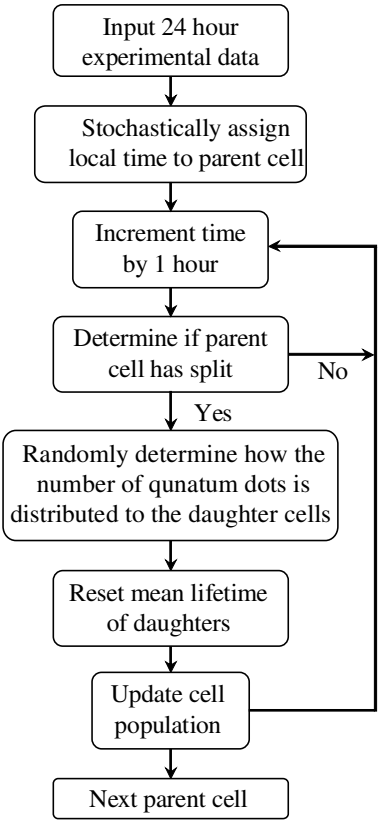


Fig. 3. Flowchart indicating the main steps of the stochastic cell-splitting algorithm

However, if the split does occur, the algorithm determines how the embedded quantum dots are distributed from the parent cell to the two daughter cells. Once more this choice is stochastically driven; selection is determined by a normal distribution centered on the mean splitting rate, μ which has an associated standard deviation, σ Both these parameters are numerically determined by means of genetic algorithm, the details of which are elucidated in the following paragraph.

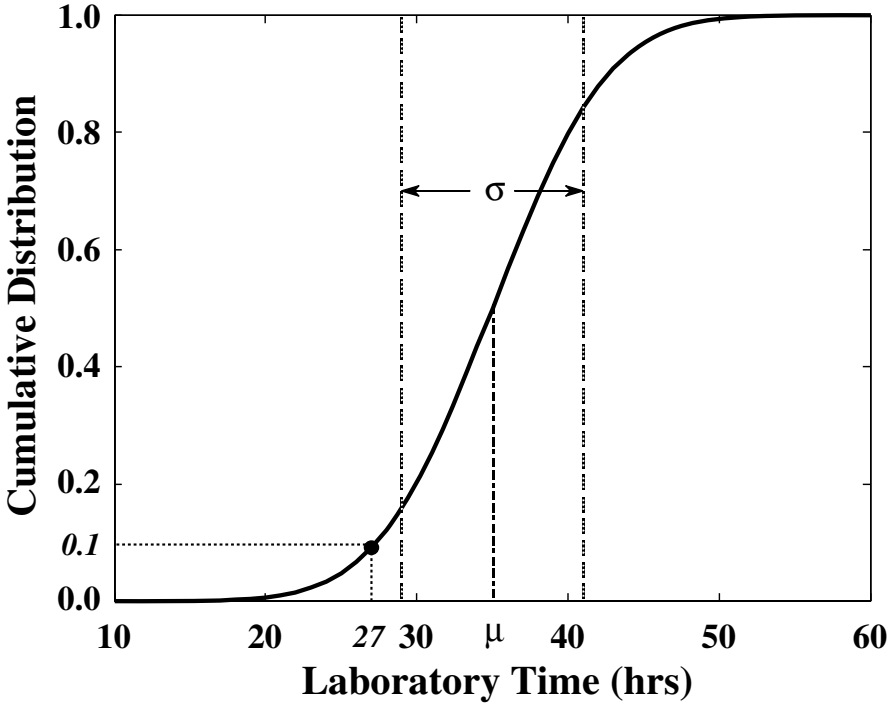


Fig. 4. Cumulative distribution associated with a cell with an initial cycle time of 12.5 hours. Plot indicates that at a laboratory time of 27 hours the cell has a 10% chance of splitting.

A modified random normal distribution routine is used to generate these values for each split cell. This routine efficiently calculates of values μonly in the unit interval [0 1]. Values generated by standard random normal distribution can lie outside this range. They can frequently occur due to the random sampling process, particularly when $\mu$is close to the extremities of the interval and $\sigma$is relatively large; in this application such values are meaningless and must be disregarded. Once the daughter cells have been assigned their respective quantum dot population, the algorithm resets their cycle time equal to their parents plus the experimentally deduced IMT. This action ensures that the probability of two newly formed daughter cells splitting again in the immediate future is small. The final stage of the algorithm simply stores both daughter and the initial parent cells yet to split in the first hour in the laboratory frame of reference. Thus, the total population is now > 10^4. Both the laboratory and cycle

time of the cells is incremented by 1 hour and a new population of 10^4 out of the augmented populace is randomly chosen to sample using the above methods. This process is repeated until, in this particular case the laboratory time reaches 72 hours.

As mentioned in the section outline, a genetic algorithm is employed to fit both the mean splitting value of the parent cells and also the associated standard deviation. A flow chart depicting the main steps of the genetic algorithm is shown below in figure 5.

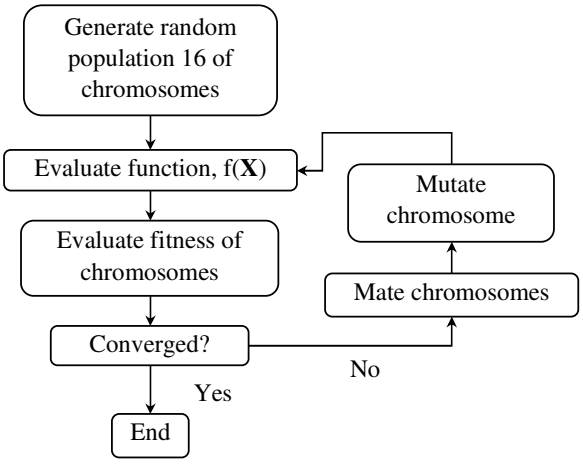


Fig 5. Flowchart of the genetic algorithm used to determine how the quantum dots are distributed from parent to daughter cells

As is typical of such algorithms, an initial population of chromosomes is randomly generated to span the parameter space, 0 to 1 for both μ and σ variables. In this particular case a population of 10 chromosomes was sufficient to ensure convergence of the algorithm. Each chromosome consisted of 16 genes, split evenly between the two minimization variables. Each chromosome's value for μ and σ was evaluated by running through the splitting procedure outlined in Figure 3, and is depicted by the function f(**X**) in Figure 5, where **X** is a vector whose components are the minimization variables. The fitness of these values was next determined by calculating the Euclidean norm of the difference between the experimental and simulated data over the entire intensity range at 72 hours. Although, the simulated data does not produce a fluorescent signal, but rather a number detailing the number of quantum dots per cell, a meaningful comparison between the experimental and simulated data can be made on the supposition that florescence is proportional to cell dot density. The fitness of the chromosome population is then evaluated by normalizing the calculated norms. If the fitness of these chromosomes does not fit the desired convergence criteria, the chromosomes are mated according to fitness and elitism (11) and a new-population born. Also, in each generation there is a small probability that each new-population chromosome undergoes a random mutation. The new-generation of chromosomes is again evaluated in the manner above until a

suitable convergence criterion is achieved; i.e. the magnitude of the minimized parameters varied by less than 5% across the chromosome population.

The genetic algorithm located strong minima for the parameters μand σat values of 0.76 and 0.06 respectively. Figure 6 above displays both the experimental and simulated quantum dot fluorescence intensity at 24, 48 and 72 hours, plots (a) and (b) respectively, when using the above values to describe how the quantum dots embedded within the parent are passed to their daughters.

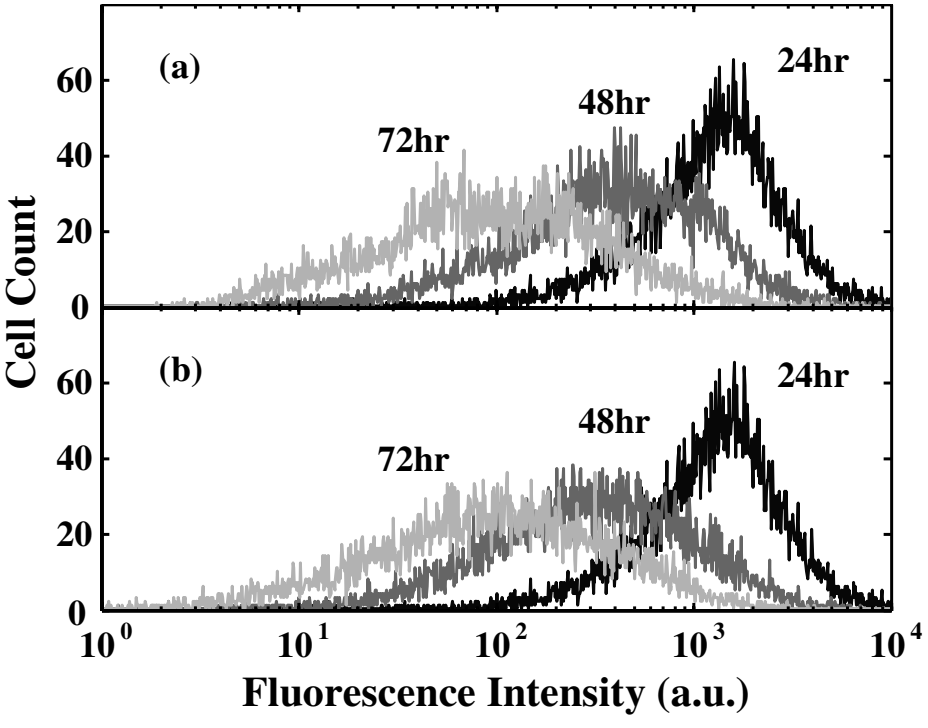


Fig. 6. (a) Experimental and (b) simulated quantum dot fluorescence intensity histograms taken at 24 hour intervals following take-up

4 Summary

We have reported on experimental methods that enable lineage tracking of quantum dot fluorophores within large cell populations over multiple generations. To complement these experimental investigations, a numerical model that mimics the cell division behavior has been developed. By utilizing a genetic algorithm in conjunction with the numerical model we have been able to achieve favorable fits of the theoretically predicted quantum dot distributions with that measured experimentally. These fits indicate that dot fluorescence is divided with a stochastic variation about an asymmetric mean split ratio of 74:26% (12) with a standard deviation of 6%.

Acknowledgments. The authors acknowledge grant support from the Research Councils UK Basic Technology Research Programme (GR/S23483) and the Biotechnology and Biological Sciences Research Council (75/E19292).

References

1. Dumitriu, I.E., Mohr, W., Kolowos, W., Kern, P., Kalden, J.R., Herrmann, M.: 5,6-Carboxyfluorescein Diacetate Succinimidyl Ester-Labeled apoptotic and necrotic as well as detergent-treated cells can be traced in composite cell samples. Analytical Biochemistry 299, 247–252 (2001)
2. Hasbold, J., Hodgkin, P.D.: Flow cytometric cell division tracking using nuclei. Cytometry 40, 230–237 (2000)
3. Bernard, S., Pujo-Menjouet, L., Mackey, M.C.: Analysis of cell kinetics using a cell division marker: Mathematical modeling of experimental data. Biophysical Journal 84, 3414–3424 (2003)
4. Alivisatos, A.P.: Semiconductor clusters, nanocrystals, and quantum dots. Science 271, 933–937 (1996)
5. Pinaud, F., Michalet, X., Bentolila, L.A., Tsay, J.M., Doose, S., Li, J.J., Iyer, G., Weiss, S.: Advances in fluorescence imaging with quantum dot bio-probes. Biomaterials 27, 1679–1687 (2006)
6. Jyoti, K., Sanford, M.S.: Potentials and pitfalls of fluorescent quantum dots for biological imaging. Trends in Cell Biology 14, 497–504 (2005)
7. Hines, M.A., Guyot-Sionnest, P.: Synthesis and characterization of strongly luminescing ZnS-Capped CdSe nanocrystals. J. Phys. Chem. 100, 468–471 (1996)
8. Dabbousi, B.O., Rodriguez-Viejo, J., Mikulec, F.V., Heine, J.R., Mattoussi, H., Ober, R., Jensen, K.F., Bawendi, M.G. (CdSe)ZnS core-shell quantum dots: synthesis and characterization of a size series of highly luminescent nanocrystallites. J. Phys. Chem. 101, 9463–9475 (1997)
9. Marquez, N., Chappell, S.C., Sansom, O.J., Clarke, A.R., Court, J., Errington, R.J., Smith, P.J.: Single cell tracking reveals that Msh2 is a key component of an early-acting DNA damage-activated G2 checkpoint. Oncogen 22, 7642–7648 (2003)
10. Errington, R.J., Ameer-Beg, S.M., Vojnovic, B., Patterson, L.H., Zloh, M., Smith, P.J.: Advanced microscopy solutions for monitoring the kinetics and dynamics of drug-DNA targeting in living cells. Adv. Drug. Deliv. Rev. 57, 153–167 (2005)
11. Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory, 1st edn. MIT Press, London (1999)
12. Njoh, K.L., Summers, H.D., Errington, R.J., Brown, M.R., Chappell, S.C., Rees, P., Matthews, D.R., Wiltshire, M., Smith, P.J.: Analysis of cell cycle perturbation using Quantum Dot fluorescence. July 2007, To be submitted to Journal of Biophysics (In preparation)

A Formal and Integrated Framework to Simulate Evolution of Biological Pathways

Lorenzo Dematté[1,2], Corrado Priami[1,2],
Alessandro Romanel[1,2], and Orkun Soyer[1]

[1] The Microsoft Research - University of Trento Centre for Computational and Systems Biology
[2] Dipartimento di Informatica e Telecomunicazioni, Università di Trento
{dematte,priami,romanel,soyer}@cosbi.eu

Abstract. This paper presents a formal approach to the study of evolution of biological pathways. The basic idea is to use the Beta Workbench to model and simulate pathway in connection with evolutionary algorithms to implement mutations. A fitness function is used to select individuals at any generation. The feasibility of the approach is demonstrated with a simple example.

1 Introduction

Biological pathways are studied using several methods, including but not limited to analysis of their single components, reconstruction of their series of chemical reactions and simulation of their dynamics.

Many approaches include modelling techniques to represent and analyse dynamics of complex pathways. In particular, *process calculi* have been recently applied to model and simulate biological systems [1]. The main activities up to now have been oriented to define new primitives and to show their feasibility for describing biological phenomena. Some simulation engines have been developed especially based on the stochastic π-calculus [2] like SPiM [3] or BIOSPI [4], and recently on an extension of the Beta-binders language [5].[1]

Recently, there is an interest in using *evolutionary approaches* to study pathways. Understanding how pathways emerged during evolution can help us to understand their basic properties, such as the role of complexity and the importance of topology and feedback loops. Existing approaches to study evolution are commonly based on comparative genomics or proteomics and on phylogenetic analysis [6,7]. These studies compare pathways from different organisms to see how evolution affects the internal structure of the network of interactions.

An alternative approach is to simulate evolution *in silico*, using an algorithm that mimic this process, as in [8,9,10,11]. However, these approaches use ad-hoc tools and representations of pathway dynamics, while current available tools to model and simulate pathway dynamics, discussed above, do not allow for evolutionary simulations.

[1] Available at url `http://www.cosbi.eu/Rpty_Soft_BetaWB.php`

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 106–120, 2007.

On the other hand, tools that mimic evolution are also known in the Computer Science domain. However, these tools and algorithms are designed for specific tasks, such as machine learning and optimization or search problems (as pioneered in 1966 by Fogel [12]), and therefore lose the strict connection with biology and they are no more adequate to study biological evolution in a realistic way.

Here, we aim at developing a specific framework to allow straightforward study of pathway evolution based on the Beta Workbench (BetaWB) [5]. The great flexibility of the Beta-binders process calculus in the definition of the structure of proteins allows us to introduce interesting primitives for mutations used to build domain-based interaction and mutation models. Moreover, pathway dynamics can be easily modelled, and the interactions of emerged pathways analysed. Our formal framework aims at extending the features of the BetaWB to obtain a full integration with the existing software environment. In this paper we describe the novelty of the approach unravelling the formal theory on which it is based, and we show the feasibility of the proposed solution to study evolution of pathways.

The paper is organised as follows. In the next section we briefly recall the language and the software tool BetaWB on top of which we built our work for simulating the evolution of biological pathways. The model we used to describe the general structure of signalling pathways is also introduced and explained. Sect. 3 discusses the evolutionary framework, the integration of the BetaWB simulation engine within an evolutionary algorithm and how mutations are performed on the model. An example of application of the whole framework is given in Sect. 4, and conclusions and future work directions close the paper.

2 Description of Pathway Dynamics

To study biological evolution of pathways, we need to describe pathway dynamics and an algorithm to simulate pathway evolution from a generation to the next one. We start describing the modeling and simulation tool.

2.1 The Beta Workbench

The Beta Workbench is a software framework for modelling and simulating biological processes [5]. It incorporates a language, a compiler to a stochastic abstract machine, the execution environment and some graphical interface components. The BetaWB language is based on β-binders [13], a process calculus developed to represent the interactions between biological entities.

The Language. A BetaWB program, called also β-system, is a tuple $Z{=}\langle B, E, \xi\rangle$ made up of a *bio-process* B, a list of *events* E and *ambient* ξ. The bio-process B intuitively represents the structure of the system, that is a set of entities interacting in the same environment, E represents the list of possible events enabled on the system and the ambient ξ contains information about the environment, like the set T of the considered *types* (ranged over by Δ, Γ_0, Σ', $\cdots$), a function

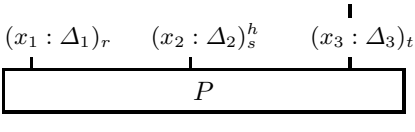
$\rho : \mathcal{N} \to \mathbb{R}$ that associates stochastic rates[2] to names in $\mathcal{N}$ and the function $\alpha : T^2 \to \mathbb{R}^3$, which describes the affinity relation between couples of types. In particular, given two types Δ and Γ, the application of $\alpha(\Delta, \Gamma)$ returns a triple of stochastic rates (r, s, t), where the value r, denoted with $\alpha_c(\Delta, \Gamma)$, represents the *complexation rate*, the value s, denoted with $\alpha_d(\Delta, \Gamma)$, represents *decomplexation rate* and the value t, denoted with $\alpha_i(\Delta, \Gamma)$, represents the *inter-communication rate* of the two entities exposing the types Δ and Δ.

The syntax of the BetaWB language is the following.

$$
\begin{array}{lll}
B & ::= & \mathsf{Nil} \mid \boldsymbol{B}[P] \mid B||B \\
\boldsymbol{B} & ::= & \widehat{\beta}(x, r, \Delta) \mid \widehat{\beta}(x, r, \Delta)\boldsymbol{B} \\
\widehat{\beta} & ::= & \beta \mid \beta^h \mid \beta^c \\
P & ::= & \mathsf{nil} \mid P|P \mid !\pi.P \mid M \\
M & ::= & \pi.P \mid M + M \\
\pi & ::= & x(y) \mid \overline{x}\langle y \rangle \mid (\tau, r) \mid (\mathsf{die}, r) \mid (\mathsf{ch}(x, \Delta), r) \mid \\
 & & (\mathsf{hide}(x), r) \mid (\mathsf{unhide}(x), r) \mid (\mathsf{expose}(x, s, \Delta), r) \\
cond & ::= & \boldsymbol{B}[P] : r \mid |\boldsymbol{B}[P]| = n \mid \boldsymbol{B}[P], \boldsymbol{B}[P] : r \\
verb & ::= & \mathsf{new}(\boldsymbol{B}[P], n) \mid \mathsf{split}(\boldsymbol{B}[P], \boldsymbol{B}[P]) \mid \mathsf{join}(\boldsymbol{B}[P]) \mid \mathsf{delete} \\
event & ::= & \bullet \mid (cond)\ verb \\
E & ::= & event \mid event :: E
\end{array}
$$

where $x, y \in \mathcal{N}$, $n \in \mathbb{N}$ and $r \in \mathbb{R}^+ \cup \{\infty\}$ is a stochastic rate.

Processes generated by the non terminal symbol P are referred as *pi-processes*. Bio-processes(boxes) are defined as pi-processes prefixed by specialised binders $\boldsymbol{B}$ that represent interaction capabilities. An *elementary beta binder* $\widehat{\beta}$ has the form $\beta(x, r, \Gamma)$ (active), $\beta^h(x, r, \Gamma)$ (hidden) or $\beta^c(x, r, \Gamma)$ (complexed), where the name x is the subject of the beta binder and Γ represents the type of x. Although types can be any general structure for which exist a decidable equality relation, without loss of generality we assume here that types are names taken from a countably infinit set T such that $T \cap \mathcal{N} = \emptyset$. A bio-process B is either the deadlock beta-process *Nil* or a parallel composition of boxes $\boldsymbol{B}[P]$. A box $\boldsymbol{B}[P]$ models a single biological entity, where intuitively the beta binders list $\boldsymbol{B}$ represents the interface of the entity, while the pi-process P represents its internal structure or process-unit. Moreover, the language is provided with a graphical representation of boxes:



The pairs $x_i : \Delta_i$ represent the sites through which the box may interact with other boxes. *Types* Δ_i express the interaction capabilities at x_i. The value r represents the stochastic rate associated to the name x inside the box, h represents

[2] A stochastic rate is the single parameter defining an exponential distribution that drives the stochastic behaviour of an action. The rate ∞ is used to denote immediate actions, i.e., actions that are executed as soon as they become enabled.

the hidden status and the black line over the last beta binder represents the *complexed* status.

The dynamics of the system is formally specified through the *operational semantics* of the language available in [5], which uses a notion of structural congruence $\equiv$. Intuitively, that two β-systems $Z{=}\langle B, E, \xi\rangle$ and $Z'{=}\langle B', E', \xi'\rangle$ are structurally congruent ($Z \equiv Z'$), if their bio-processes B and B' and their list of events E and E' are identical up to structure ($B \equiv_b B'$ and $E \equiv_e E'$) and their ambients are equal ($\xi{=}\xi'$). Moreover, two boxes representing interacting entities, are considered of the same species only if they are structurally congruent. Given a β-system, its temporal evolution is described by three types of actions.

Monomolecular actions describe the dynamics of single boxes. More precisely, an *intra-boxes communication* allows components to interact within the same box, the *expose* action adds a new site of interaction to the interface of the box containing the expose, the *change* action modifies the type of an interaction site, *hide* and *unhide* actions make respectively invisible and visible an interaction site. Finally, the *die* action eliminates the box that performs the action and, recursively, all the boxes directly or indirectly complexed with them.

Bimolecular actions describe interactions that involves two boxes. The *complex* operation creates a dedicated communication binding between boxes over compatible and unhidden elementary beta binders, while the *decomplex* operation destroys an already existing dedicated binding. The stochastic rates associated to complex and decomplex operations are, respectively, the complexation and decomplexation rates derived from the affinity function. In the example, the complexation rate is $\alpha_c(\Delta, \Gamma)$, while the decomplexation rate is $\alpha_d(\Delta, \Gamma)$. The information about the existing dedicated bindings is maintained in the environment. The last bimolecular action is the *inter-boxes communication*, which enables interaction between boxes over compatible and unhidden elementary beta binders. Suppose Δ and Γ be the types associated to the involved elementary beta binders. If $\alpha_c(\Delta, \Gamma) > 0$, then the *inter-communication* is enabled, with rate $\alpha_i(\Delta, \Gamma)$, only after a dedicated communication binding, over the involved beta binders, has been created by a *complex* operation. Otherwise, the *inter-boxes communication* is simply enabled with rate $\alpha_i(\Delta, \Gamma)$.

An **Event** is the composition of a condition *cond* and an action *verb* and it is triggered only when the condition associated with the event is satisfied. Events were designed to substitute the join and split actions of the original beta-binders language with a more efficient construct. They can be considered as global rules on the system which can substitute, create and delete boxes. In particular, *verbs* originate five types of events: *join*, which substitutes two boxes with single ones; *split*, which substitutes a box with two boxes; *new*, which introduces a specified number of instances of a box; *delete*, which eliminates boxes. These different types of events are triggered three kinds of *conditions*: conditions on the presence of a bioprocess, conditions on cardinality of a bioprocess, and conditions on time.

Definition 1. *The BetaWB* Stochastic Transition System *(*STS*) is referred as* $\mathcal{S} = (\mathcal{Z}, \xrightarrow{r}_s, Z_0)$*, where* $\mathcal{Z}$ *is the set of* β*-systems,* $Z_0 \in \mathcal{Z}$ *is the initial* β*-system and* $\xrightarrow{r}_s \subseteq \mathcal{Z} \times \mathbb{R} \times \mathcal{Z}$ *is the stochastic transition relation, where* r *is a stochastic*

rate constant and is derived using information in the syntax and in the ambient of the β-system.

For a more detailed and formal description of the language, its operational semantics and the laws of its structural congruence relation we refer the reader to [5].

2.2 A Compositional Model For Signalling Pathways

A *signalling pathway* is any biological process that converts one kind of signal or stimulus into another; this conversion is also called *signal transduction*. In general, a signalling pathway results in a composition or cascade of biochemical reactions that are carried out by proteins and linked through second messengers. Biological signal transduction allows a cell or organism to sense its environment and react accordingly. Typically, a signalling pathway has one (or more) input, represented by any environmental stimulus, and one (or more) output, represented by an active protein. In the BetaWB language we represent a protein

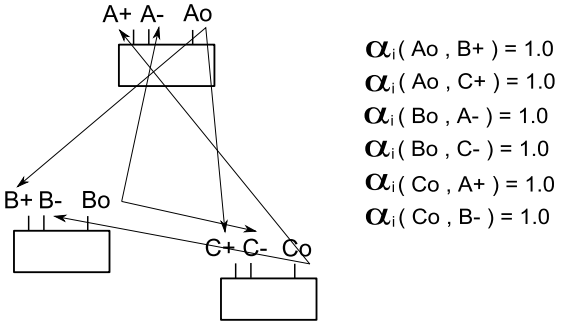


Fig. 1. Our representation of protein interaction through sensing and effecting domains. Arrows indicate compatibility of the types, according to the α function specification reported on the right of the figure.

as a biological entity composed by a set of *sensing domains*, a set of *effecting domains* and an *internal structure*. Sensing domains are the places where the protein receives signals, effecting domains are the places that a protein uses for propagating signals, and the internal structure codifies for the mechanism that transforms an input signal into a protein conformational change, which can result in the activation or deactivation of another domain. This is inspired by the available knowledge of protein structure and function (see for example [14]).

As explained in Sec. 2.1, the Beta Workbench is a general framework for modeling and simulating biological processes; each biological entity is modelled with a box, which is a composition of an interface and an internal process unit. This gives an effective way for modeling proteins by decomposing the domains of interaction and the internal structure into two different constructs. Moreover, the

compositional nature of the language allows us to design and apply mutations on biological entities in an effective, simple and intuitive way.

In this application of evolutionary algorithms to β-systems we designed a general methodology for modeling proteins by providing patterns for modeling interaction domains and internal structures. Domains are represented using beta binders ($A+$ and $A-$), interaction sites with an affinity. As shown in Fig. 1 (where we omit subjects and rates of beta binders for the sake of readability), the *sensing domain* is represented by a pair of binders, one for receiving a signal of *activation* (e.g. phosphorylation) and the other for receiving a signal of *deactivation* (e.g. dephosphorylation) sent to the protein. The *effecting domain* (Ao) is instead used to communicate, and so to activate or inhibit, other proteins. Pattern of pi-processes and boxes for modeling proteins with single sensing and effecting domains are the following:

$$
\begin{array}{ll}
ps_1 & \triangleq act(\iota).(\mathsf{unhide}(o),\infty).deact(\iota).(\mathsf{hide}(o),\infty).\overline{x}\langle\omega\rangle.\mathsf{nil} \\
pr_1 & \triangleq !x(\iota).ps_1 \mid ps_1 \\
pss_2 & \triangleq deact(\iota).(\mathsf{hide}(o),\infty).\overline{xp}\langle\omega\rangle.\mathsf{nil} \\
ps_2 & \triangleq !xp(\iota).(act(\iota).(\mathsf{unhide}(o),\infty).pss_2 + deact(\iota).\overline{x}\langle\omega\rangle.\mathsf{nil}) \\
pr_2 & \triangleq !x(\iota).act(\iota).\overline{xp}\langle\omega\rangle.\mathsf{nil} \mid act(\iota).\overline{xp}\langle\omega\rangle.\mathsf{nil}
\end{array}
$$

$$
\begin{array}{ll}
Adeact_1 & \triangleq \beta^h(o,r,\Delta^o)\beta(act,r,\Delta^+)\beta(deact,r,\Delta^-) \\
 & \quad [\ pr_1 \mid !\overline{o}\langle\omega\rangle\ ] \\
Aact_1 & \triangleq \beta(o,r,\Delta^o)\beta(act,r,\Delta^+)\beta(deact,r,\Delta^-) \\
 & \quad [\ !x(\iota).ps_1 \mid deact(\iota).(\mathsf{hide}(o),\infty).\overline{x}\langle\omega\rangle.\mathsf{nil} \mid !\overline{o}\langle\omega\rangle\ ] \\
Adeact_2 & \triangleq \beta^h(o,r,\Delta^o)\beta(act,r,\Delta^+)\beta(deact,r,\Delta^-) \\
 & \quad [\ ps_2 \mid pr_2 \mid !\overline{o}\langle\omega\rangle\ ] \\
Aact_2 & \triangleq \beta(o,r,\Delta^o)\beta(act,r,\Delta^+)\beta(deact,r,\Delta^-) \\
 & \quad [\ !x(\iota).act(\iota).\overline{xp}\langle\omega\rangle.\mathsf{nil} \mid pss_2 \mid ps_2 \mid !\overline{o}\langle\omega\rangle\ ]
\end{array}
$$

where $\rho(x) = \rho(xp) = \infty$. The boxes $Adeact_1$ and $Aact_1$, which uses the pi-processes ps_1 and pr_1, represent respectively the inactive and active state of a protein which can be activated by a single external signal. In particular, if the box $Adeact_1$ executes an *inter-boxes communication* through the elementary beta binder $\beta(act,r,\Delta^+)$, the action $act(\iota)$ in $Adeact_1$ is consumed and immediately also the action $(\mathsf{unhide}(o),\infty)$ is consumed (because its rate is ∞). The obtained box is structurally congruent to $Aact_1$ and hence the protein has reached its active form, where the elementary beta binder $\beta(o,r,\Delta^o)$ is now unhidden and the box can execute inter-communications through it. Now, if the box $Aact_1$ executes an *inter-boxes communication* through the elementary beta binder $\beta(deact,r,\Delta^-)$, the reverse mechanism is executed and the protein returns back in its inactive form $Adeact_1$.

The boxes $Adeact_2$ and $Aact_2$, which uses the pi-processes ps_2, pss_2 and pr_2, represent respectively the inactive and active state of a protein which can be activated by receiving a signal twice. The activation and deactivation mechanism is similar to the one described for single signal activation.

Obviously these patterns can be easily extended for modelling proteins with more than one sensing and effecting domains and for modeling mechanisms of activation based on the reception of more than two external signals.

3 Evolutionary Framework

We design a framework for simulating the evolution of pathways *in silico*. Evolution proceeds through selection acting on the variance generated by random mutation events. Individuals replicate in proportion to their performance, referred to as *fitness*. This process can be modelled as shown in Tab. 1. This

Table 1. Generic EvolutionAlgorithm

```
EvolutionAlgorithm ():
  Population := GenerateInitialPopulation();
  for i = 0 to generations do
    for each Individual in Population do
      output := Simulate(Individual);
      fitnesses[Individual] := ComputeFitness(output);
    NewPopulation := ReplicateAndMutate(fitnesses, Population);
    Population := NewPopulation;
```

algorithm differs slightly from the generic evolutionary algorithms used in computer science, being more closer to real biological observations made for the asexual reproduction of organisms.

There are four main procedures in the algorithm:

- **GenerateInitialPopulation:** the initial population can be generated randomly, from a predefined pathway configuration to be used as a starting point, or it can be a pathway with no interactions. All the individuals in the initial population can be equal at the beginning, as they will be differentiated later by the mutation phase.
- **Simulate:** each individual in the population is simulated separately using the BetaWB stochastic simulator, introduced in Sec. 2.1.
- **ComputeFitness:** the output of the simulation is used to compute the fitness value of the current individual. Note that the fitness value is problem-dependent; for an example, refer to Sec. 4.
- **ReplicateAndMutate:** this is the most important part of the algorithm; like in a real environment, individuals with the highest fitness values are more likely to survive, replicate and produce a progeny that resembles them, being not, however, completely equal to them.

The ReplicateAndMutate algorithm (Tab. 2) creates a new population with the same number of individuals of the current generation, using as a base the current individuals. At each step it chooses one individual, with probability proportional to its fitness (ChooseOneIndividual in the code above). This

Table 2. The REPLICATEANDMUTATE algorithm

```
REPLICATEANDMUTATE (fitnesses, Population):
  for  i = 0  to  i < Population.Size  do
    Individual := ChooseOneIndividual(Population, fitnesses);
    for each  Protein  in  Individual.Proteins  do
      if  Random() < DuplicationProbabily  then
        Protein2 := Protein.Duplicate();
        Individual.Proteins.Add(Protein2);
      for each  Domain  in  Protein.Domains  do
        if  Random() < MutationProbability  then
          MutationType := GetRandomMutation();
          if  IsMutationFeasible(Domain, MutationType)  then
            Domain2 := Individual.PickCompatibleDomain(Domain, MutationType);
            Individual.Mutate(Domain, Domain2, MutationType);
      NewPopulation.Add(Individual);
    return  NewPopulation;
```

is achieved by constructing a cumulative probability array a from the *fitness* array, generating a random number in the range $0...a[Population.Size]$, and then finding the index into which the random number falls.

The selected individual will replicate and pass to the next generation. During the replication, each protein in the "genome" of the individual is given the chance to mutate, according to a probability.

A mutation is selected among all the possible types by the GETRANDOMMUTATION function, and this mutation is applied. Finally the individual, that can be equal to its predecessor or can be mutated, is added to the new population. We now define in more detail the kind of mutations that we consider in our framework.

3.1 Mutations

The different types of mutations we consider are based on real biological processes where mutations can happen at DNA and protein level. These ultimately effect pathway dynamics. For example, point mutations in a DNA sequence can change the protein amino-acid sequence, leading to changes in its tertiary structure with implications on the affinity of this protein with other proteins or substrates. Similarly, events at DNA level as gene duplication or domain shuffling can alter pathway structure and dynamics.

A computer program that wants to mimic evolution of a specie have to implement random mutations in individual during replication as well. Here, we can easily implement these molecular processes using the domain and pathway model we discussed in Sect. 2.2.

We will take as an example the three-protein pathway represented in Fig. 2(a) and we will illustrate how different mutations are possible using the BetaWB language.

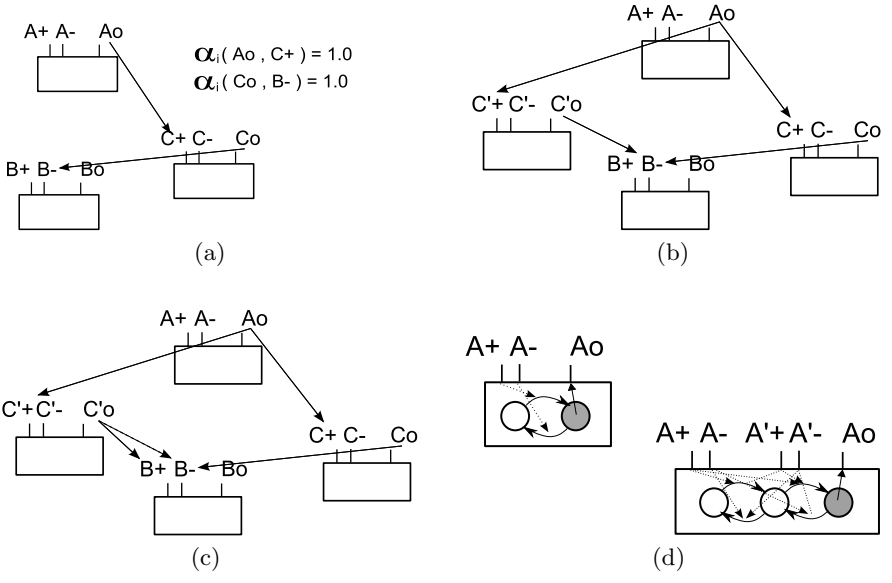


Fig. 2. Different kinds of mutations: in (a) the initial configuration, displaying the α function as a list of tuples; in (b) duplication of protein C followed by mutation of domain Δ^{Co} in (c). Finally, (d) displays how the internal structure could change to accommodate the duplication of a domain.

Duplication and Deletion of Proteins: Gene duplication at DNA level was implemented in our framework with a duplication of the bio-processes associated to the protein the gene codifies for. The new bio-process will have the same internal structure and the same binder subjects, while types of binders will be new but will have the same interaction capabilities. This is achieved by copying the affinities of the original types. Duplication of types is needed because subsequent mutations on one of the binders of the duplicated protein must not affect the original one. Furthermore, since the new protein is a new separated entity, it must not be structural equivalent to the original one. Following the model presented in Sec. 2.2, the bio-process for protein C

$$Cdeact_1 \triangleq \beta^h(o, r, \Delta^{Co})\beta(act, r, \Delta^{C+})\beta(deact, r, \Delta^{C-})[\ pr_1 \mid !\overline{o}\langle\epsilon\rangle\ ]$$

will be duplicated to

$$Cdeact_1 \triangleq \beta^h(o, r, \Delta^{Co})\beta(act, r, \Delta^{C+})\beta(deact, r, \Delta^{C-})[\ pr_1 \mid !\overline{o}\langle\epsilon\rangle\ ]$$

$$C'deact_1 \triangleq \beta^h(o, r, \Delta^{C'o})\beta(act, r, \Delta^{C'+})\beta(deact, r, \Delta^{C'-})[\ pr_1 \mid !\overline{o}\langle\epsilon\rangle\ ]$$

The same duplication will be done also for the bio-process representing the active forms of the protein. Deletion can be implemented removing the associated bio-process, or setting its cardinality to zero.

Mutation of Domains: Point mutations in DNA can change the protein amino-acid sequence, and consequently lead to the mutation of a domain and to changes in the interaction capabilities of the protein to which it belongs. In our formalism, this is achieved by changing the α function on the two domains that take part in the interaction. More specifically, the mutation on a domain can be a *change of interaction*, for which we modify the affinity adding a number sampled from a normal distribution with mean zero and a little variance, an *addition of an interaction* between two domains $d1$ and $d2$, modelled as the addition of an affinity $\alpha(d1, d2) = x$, whit $x > 0$, and finally a *removal of an interaction*, between two domains $d1$ and $d2$, setting $\alpha(d1, d2) = 0$. For example, the mutation on domain oC that can be observed in Fig. 2(c) is obtained by changing the α function from $\alpha(Co, B-) = 1.0$ to $\alpha(Co, B-) = 0.0$,$\alpha(Co, B+) = 0.9$

Duplication and Deletion of Domains: The last possible mutation, domain duplication or deletion due to DNA shuffling, is more complex as it requires modification of the internal behaviour in response to stimuli. Duplicating or removing domains can be easily done acting on the binders list and on the affinity function α; however, for these domains to act as sensing or effecting domains in cooperation or in antagonism with the existing ones, the internal behaviour of the process must also be changed. We devised several possible modifications of the behaviour when a domain is added, such as require communication on all the sensing domains before activating the protein (double phosphorylation, as in Fig. 2(d)), different patterns of activation when more than an effecting domain exists, and so on. All these modifications can be done manipulating the abstract syntax tree of the bio-process, duplicating parts of it in a regular way. In Fig. 3.1, for example, it is shown how it is possible to manipulate a bioprocess to transform a protein activated (or deactivated) by a single phosphorylation

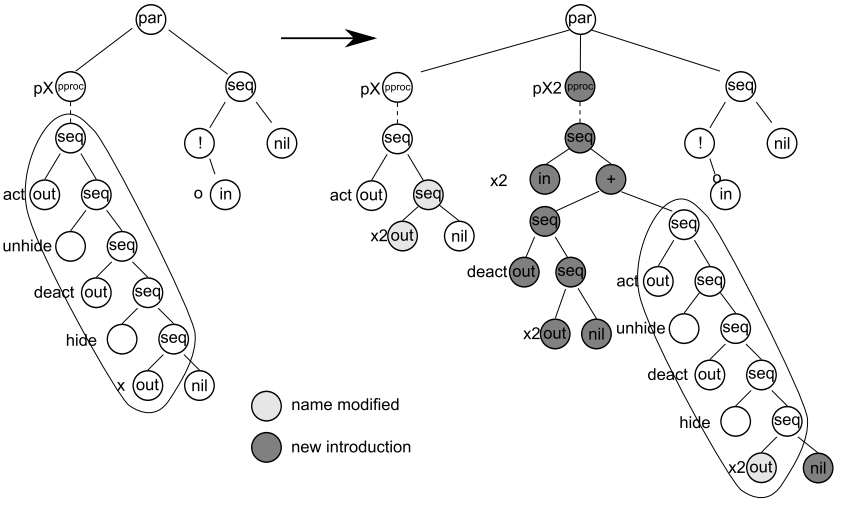


Fig. 3. Program transformation for the modification of a sensing domain

into a protein that is activated (or deactivated) by a double phosphorylation, encoding an intermediate step of "half-activation". However, not all of the possible combinations of transformations have a correspondent biological meaning; a more rigorous investigation on these kind of mutations is to be done before including them in our framework and tool.

3.2 Measure of Fitness

The measure of fitness is problem dependent: it varies with the kind of pathways, with the characteristics a scientist wants to investigate, and so on. This measure can be done in various ways, including stability analysis, integration of the signal, measure of the derivative. We will illustrate in our example how fitness can be computed using integration of a response.

4 An Example: MAPK Cascade

The mitogen-activated protein kinase cascade (MAPK cascade) is a series of three protein kinases, which is responsible for cell response to growth factors. In [15], an model for the MAPK cascade was presented (Fig.4) and analysed using ODEs; the cascade was shown to perform the function of an ultra-sensitive switch and the response curves were shown to be steeply sigmoidal. A process calculi

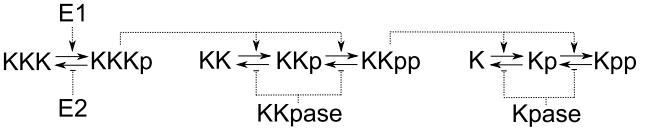


Fig. 4. MAPK cascade as described in [15]. KKK denotes $MAPKKK$, KK denotes $MAPKK$ and K denotes $MAPK$. The signal $E1$ transforms KKK to $KKKp$, which in turn transforms KK to KKp to $KKpp$, which in turn transforms K to Kp to Kpp. In particular, when an input $E1$ is added, the output of Kpp increases rapidly. The transformations in the reverse direction are the result of the signal $E2$, the $KKpase$ and the $Kpase$. In particular, by removing the signal $E1$, the output level of Kpp revert back to zero.

based analysis of the MAPK cascade was presented in [16]. For simplicity, in this paper we rely on a simplified version of the model, where all the enzymatic reactions of the form:

$$E + S \rightleftharpoons^{K_{ES}}_{K_{ES}^{-1}} ES \multimap^{K_{EP}} EP \multimap^{\infty} E + P$$

are substituted with simplest reactions of the form:

$$E + S \multimap^{K_{EP}} E + P$$

Using the design patterns presented in 2, a β-system for the MAPK cascade has been developed. Following [16], we set all the reaction rates to a nominal

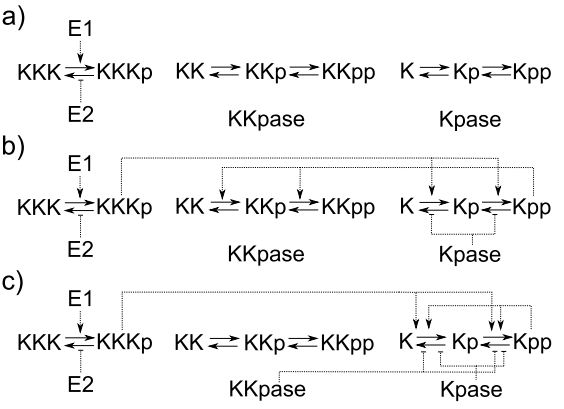


Fig. 5. (a) Basic individual of the initial configuration. Only signals $E1$ and $E1$ are enabled. (b) A particular individual we obtained, with a "reverse" cascade. (c) A simil-mapk we obtained, with a good fitness value.

value of 1.0 and we initialize the system with two of $E1$, $E2$, $KKPase$ and $KPase$, 20 of KKK and 200 of KK and K. Simulating the MAPK β-system with the BetaWB simulator, similar response profiles (modulo timescale) were observed for the output of Kpp with respect to the model presented in [15], despite the differences in the simulation parameters; the system still behaves as an ultra-sensitive switch.

We use this simplified MAPK cascade β-system as a toy model for testing our evolutionary framework. In particular, we want to analyse the evolution of a population according to a fitness function which captures the essential behaviour of our MAPK cascade model.

In detail, we generate an initial population of 500 individuals containing the pathway shown in Fig.5a. This pathway resembles the observed MAPK cascade, in that it contains three kinases, two phosphatases and two signalling molecules. However, it lacks any interactions among these. In other words, we consider an ancestral organism that possessed all these proteins but lacked a signalling system similar to the MAPK cascade as observed today. The dynamic of each individual is then simulated; we run each individual for 7000 simulation steps and we remove the signal $E1$ at the step 1500 using a time-triggered *delete* event, introduced in Sec. 2.1. Using the output of the simulation, we then measure for each individual the corresponding fitness. The fitness function we implemented measures how rapidly the output of Kpp increases, how much the output of Kpp persists after removing the signal $E1$ and how rapidly the output of Kpp returns back to zero. Let $out = \{n_0, n_1, ..., n_{7000}\}$ be the tuple representing the Kpp dynamics in time of an individual, then the fitness for out is formally computed by the following formula:

$$fitness(out) = \mu + \left(\frac{\sum_{j=i_1}^{e1} n_j}{Kpp_M * (e1 - i1)} - \left(\gamma * \frac{\sum_{j=i_2}^{e2} n_j}{Kpp_M * (e2 - i2)} \right) \right)$$

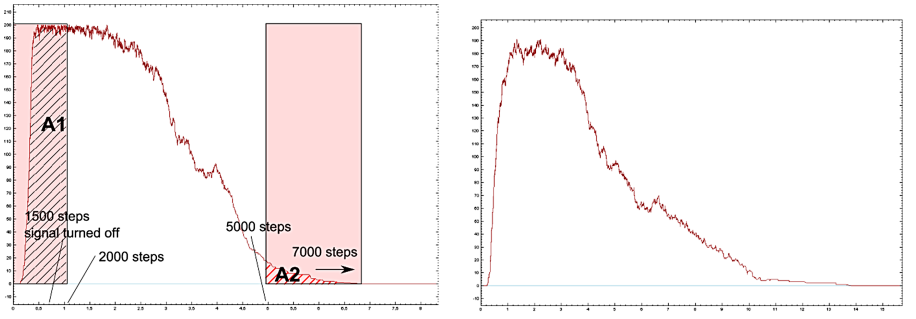


Fig. 6. (a) Time course of the Kpp concentration over the simulation time, superimposed to the integral areas for the fitness function we implemented. The fitness parameters are $i_1 = 0$, $e_1 = 2000$, $i_2 = 5000$ and $e_2 = 7000$. (b) An individual with high fitness.

The two sums, that we denote respectively with $A1$ and $A2$, represent discrete integrals and are normalised with respect to their possible maximum values (see Fig. 6). The values i_1, e_1, i_2 and e_2 are changeable parameters that define the boundaries for the computation of the two discrete integrals present in the formula, and the value Kpp_M represents the maximum value for the Kpp response. Moreover, μ represents the minimum fitness and γ controls the relative importance to responding to a signal and turning the response off after its removal. The reported results are for $i_1 = 0$, $e_1 = 2000$, $i_2 = 5000$, $e_2 = 7000$, $Kpp_M = 200$, $\mu = 0.1$ and $\gamma = 0.65$.

According to the algorithms presented in the previous section, the population is evolved. In this case study only *mutations of domains* are applied. Moreover, in order to maintain a biological validity for the new individuals, possible mutations are the one that satisfies the following constraints: (1) Signals $E1$ and $E2$ act only on KKK; (2) signal $E1$ and $E2$ cannot be removed; (3) a kinase can only activate other kinases or itself; (4) phosphatases are not specific but can only deactivate kinases. We iterate the evolution algorithm for 1000 generations, for different values of fitness function parameters. The dynamic for one of the obtained pathways is shown in Fig. 6(b). Examples of obtained individuals are in Fig. 5, while the variation of fitness during a simulation is depicted in Fig. 7. In particular, we do not obtain individuals with a perfect MAPK cascade pathway, but individuals in (b) and (c) are only a reordering and a protein duplication event away from it.

Moreover, we observe three different plateaus on the average fitness variation shown by the first chart in Fig. 7. The first one is obtained after pathways evolve a kinase interaction, the second one is obtained after pathways evolve addition of a phosphatase interaction, and the third one, which persists till the final generation, is obtained after addition of other two interactions -one from a kinase and one from a phosphatase-.

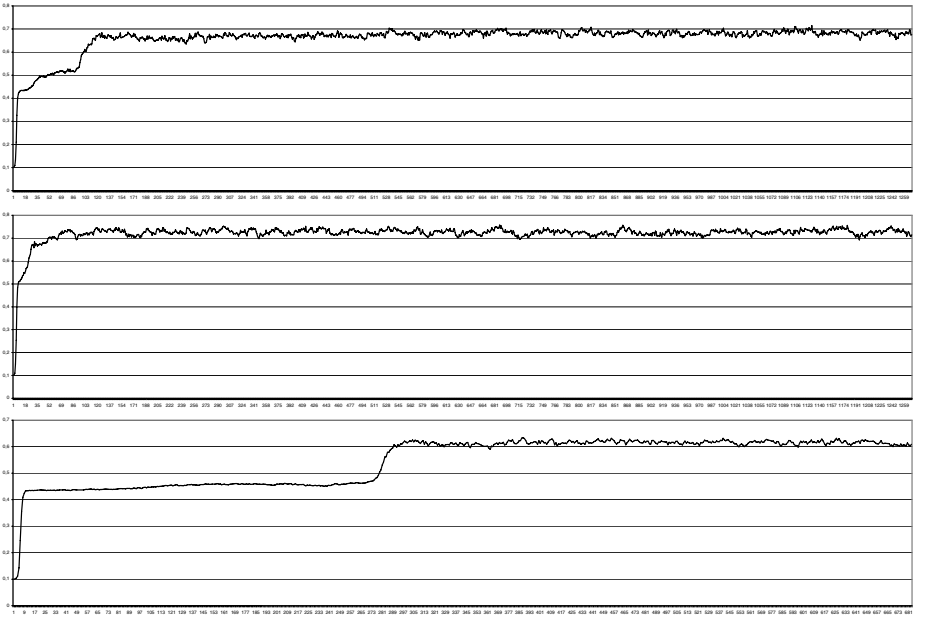


Fig. 7. The modification of fitness during three different runs of the *in silico* evolution

5 Conclusions and Future Work

In this paper we present a formal approach for simulating the evolution of pathways *in silico*. Pathway dynamics are described with the BetaWB language, a process algebra that, thanks to its biological-inspired nature, allows for modelling proteins, domains and interactions in a modular way.

We developed a modular description of signalling pathways, an evolutionary algorithm and a prototype to test our concepts. The prototype showed the potential of our approach, and the beta-binders language proved to be well-suited for this task. As presented in the small example in Sec. 4, by simulating evolution we can gain interesting insights in pathway structure and on the role of different processes.

We implemented the simpler kinds of mutations, but other interesting variations like modification of internal process structure are feasible and it will surely be a subject worth of future investigation and development.

Also, an easy to use and integrated tool for the simulation and analysis of pathway evolution under development. Finally, we plan to use the same framework also from an optimization perspective to help discovering new pathways.

References

1. Regev, A., Shapiro, E.: Cells as computation. Nature (2002)
2. Priami, C., Regev, A., Shapiro, E., Silvermann, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Inf. Process. Lett. 80(1), 25–31 (2001)

3. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. In: Bioconcur'04, ENTCS (August 2004)
4. Regev, A., Silverman, W., Shapiro, E.: Representation and simulation of biochemical processes using the pi-calculus process algebra. Pac. Symp. Biocomput. , 459–470 (2001)
5. Romanel, A., Dematté, L., Priami, C.: The beta workbench. Technical Report TR-03-2007, The Microsoft Research - University of Trento Centre for Computational and Systems Biology (2007)
6. Sharan, R., Suthram, S., Kelley, R.M., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R.M., Ideker, T.: Conserved patterns of protein interaction in multiple species. Proc. Natl. Acad. Sci. 6(102), 1974–1979 (2005)
7. Babu, M.M., Teichmann, S.A., Aravind, L.: Evolutionary dynamics of prokaryotic transcriptional regulators. J. Mol. Biol. 358, 614–633 (2006)
8. Soyer, O., Bonhoeffer, S.: Evolution of complexity in signaling pathways. PNAS 103, 16337–16342 (2006)
9. Pfeiffer, T., Schuster, S.: Game-theoretical approaches to studying the evolution of biochemical systems. Trends. Biochem. Sci. 1(30), 5–20 (2005)
10. Pfeiffer, T., Soyer, O.S., Bonhoeffer, S.: The evolution of connectivity in metabolic networks. PLoS Biol. 7(3) (2005)
11. P, P.F., Hakim, V.: Design of genetic networks with specified functions by evolution in silico. Proc. Natl. Acad. Sci. 2(101), 580–585 (2004)
12. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence through Simulated Evolution. John Wiley, New York (1966)
13. Priami, C., Quaglia, P.: Beta binders for biological interactions. In: CMSB, pp. 20–33 (2004)
14. Stock, A.M., Robinson, V.L., Goudreau, P.N.: Two-component signal transduction. Annu. Rev. Biochem. 69, 183–215 (2000)
15. Huang, C.Y., Ferrell, J.E.: Ultrasensitivity in the mitogen-activated protein kinase cascade. Proc. Natl. Acad. Sci. U S A 93(19), 10078–10083 (1996)
16. Phillips, A., Cardelli, L., Castagna, G.: A Graphical Representation for Biological Processes in the Stochastic pi-Calculus. In: Transactions on Computational Systems Biology VII. LNCS, vol. 4230, Springer, Heidelberg (2006)

Reconstruction of Mammalian Cell Cycle Regulatory Network from Microarray Data Using Stochastic Logical Networks

Bartek Wilczyński[1,2] and Jerzy Tiuryn[2]

[1] Institute of Mathematics, Polish Academy of Sciences,
Śniadeckich 8, 00-956 Warsaw, Poland
bartek@impan.gov.pl
[2] Institute of Informatics, Warsaw University,
Banacha 2, 02-089 Warsaw, Poland
tiuryn@mimuw.edu.pl
http://bioputer.mimuw.edu.pl

Abstract. We present a novel algorithm for reconstructing the topology of regulatory networks based on the Stochastic Logical Network model. Our method, by avoiding the computation of the Markov model parameters is able to reconstruct the topology of the SLN model in polynomial time instead of exponential as in previous study [29]. To test the performance of the method, we apply it to different datasets (both synthetic and experimental) covering the expression of several cell cycle regulators which have been thoroughly studied [18,11]. We compare the results of our method with the popular Dynamic Bayesian Network approach in order to quantify the ability to reconstruct true dependencies. Although both methods able to recover only a part of the true dependencies from realistic data, our method gives consistently better results than Dynamic Bayesian Networks in terms of the number of correctly reconstructed edges, sensitivity and statistical significance.

1 Introduction

Modelling and reconstruction of regulatory networks is currently considered one of the central problems in Computational Biology. Since the genomic data is readily available for many organisms we face now the problem of deciphering the regulatory code driving the expression of thousands of genes. Experimental approach to reverse engineering a genetic network can be done with remarkable results, such as the work of Davidson [7]. However the number of biological experiments required to reconstruct such a network prevents this approach from wider adoption. This leads us to consider computational approaches to understand these complex processes. Such methods, based on the analysis of expression data could be very helpful in limiting the number of required wet lab experiments by pointing out the plausible topology of regulatory dependencies.

Due to the advances in microarray technology, it is now relatively inexpensive to obtain high-throughput expression measurements. Moreover such data is now

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 121–135, 2007.

available in public databases such as Arrayexpress [20] being an invaluable source of information on the regulatory processes taking place in many organisms. Unfortunately, the inference of regulatory dependencies from microarray data on a genomic scale faces many problems stemming from the inherent properties of the microarray data: large amount of noise (coming both from the stochastic nature of the process and from measurement errors) and incomplete sampling of the expression space. Overcoming these problems is a major goal for researchers in this active field of research.

1.1 Related Work

Microarray experiments provide data in the form of snapshot images capturing the expression of many genes averaged among thousands of individual cells. If multiple microarray experiments are performed on one biological system of interest, we obtain a sample from the expression space of that system. In the case of microarray time-series when we have measurements taken from a synchronized cell colony at different time-points we may assume, that each measurement represents a state on the trajectory in the state space of our system. The theoretical work on this problem was pioneered in the REVEAL algorithm [15] using the model of synchronous Boolean networks and the measure of mutual information to infer the most likely wiring of the regulatory network. In the follow-up study [1], authors show that the reconstruction of Boolean networks from a reasonable number of experiments (~ 100) may be done reliably only for networks with limited in-degree (< 2) and under the assumption of error-free measurements. Since both of these assumptions are not true in case of real biological networks, there is a need for a more robust method of network reconstruction that would take into account the stochastic nature of the biological processes.

Friedman [12] proposed to use a method of learning Bayesian Networks as a natural way of handling the probabilistic aspects of inference from microarray data. Their approach was dedicated to the analysis of static microarray data and was later adapted by Husmeier [13] to the analysis of microarray time-series. His study used the Markov Chain Monte Carlo heuristic method for inference of Dynamic Bayesian Networks (DBN) [21,17]. This approach was later extended by Dojer et al. [9] by adapting an optimal algorithm proposed by Ott et al. [19] instead of MCMC heuristic and allowing for knockout experiments. In this work we use the algorithm proposed by Dojer [8] for DBN reconstruction, since it is the most efficient implementation currently available.

Other approach to gene network reconstruction employs Probabilistic Boolean Networks (PBN) [25]. There is a very recently published [33] method of inference for DBNs based on the Minimum Description Length (MDL) principle which promises better computational complexity for inferring DBNs using the MDL score. This method restricts the space of DBN networks to the ones that can also be modelled using PBN models. Unfortunately we could not directly compare with that method since there is no publicly available implementation, but the results should match closely those obtained by DBN approach with MDL scoring function.

All the mentioned methods assume that the expression data is qualitative rather than quantitative and use discretized expression levels for inference. We agree that it is reasonable, since the measurement errors and problems with cross array normalization and comparisons justify this assumption. It should be noted, however that there are other methods based on processing of non-discretized data such as BNRC score for Bayesian Networks [14], parameter estimation for Stochastic Differential Equations (SDE) [4] and very recent MTDC method [24]. We could not directly compare our result with these methods, since most of the data we used in this work comes from discrete models for which they are not applicable.

2 Approach

Our approach is based on our previous work on the Stochastic Logical Network model for regulatory networks [29]. Here we provide new efficient algorithm for the problem of network topology reconstruction. In this section we briefly introduce the SLN modelling and present our algorithm.

2.1 Stochastic Logical Networks

The SLN model extends the kinetic logic approach [26] by taking into account the stochastic nature of regulatory processes. In the kinetic logic approach, the Boolean gene network is viewed as a discrete version of a dynamical system with regulatory dependencies described with ordinary differential equations of the form:

$$\frac{\partial v_i}{\partial t} = -v_i \cdot \lambda_i + \mathcal{F}_i(\boldsymbol{v}(t)) \tag{1}$$

where the production rate $\mathcal{F}_i$ of i-th gene depends on the sum of products of concentrations of other genes :

$$\mathcal{F}_i(\boldsymbol{v}(t)) = \sum_{X \subseteq \{1..n\}} I_{X,i} \cdot \prod_{j \in X} sigm_k\left(v_j(t) - \theta_{j,i}\right), \tag{2}$$

where $sigm_k(x) = \frac{1}{1+e^{-kx}}$, λ_i is a degradation constant of i-th gene, $I_{X,j} \in \mathbb{R}$ is the influence of the presence of all genes from the subset X on the production of ith gene and $\theta_{i,j}$ is the activation threshold of the influence of j-th gene on i-th. Under these assumptions, the activation thresholds $\theta_{k,j}$ for $k \in 1..n$ divide the concentrations of j-th gene into at most $n+1$ intervals. Thomas [26] observed that we can treat all states that fall into the same intervals as equal since the production rates are almost constant for them. For the discrete system, the production rates from Equation (2) can be transformed into a generalized Boolean rule similar to alternative of conjunctions.

This formalism has been proved to be applicable to modelling of many known regulatory networks in different organisms [22,16,11] and it is useful for verification of the qualitative properties of a known system [3]. Unfortunately, it is

not well suited for the task of network reconstruction since, mainly due to the non-deterministic nature of the process, there are exponentially many models that can explain any given dataset [29].

The SLN model overcomes this deficiency by replacing the non-deterministic state transitions by a stochastic process. It is achieved by addition of a Gaussian noise to each of the production rates from Equation (1):

$$\frac{\partial v_i}{\partial t} = -v_i \cdot \lambda_i + \mathcal{F}_i(\boldsymbol{v}(t)) + \epsilon \partial W. \tag{3}$$

Since we do not modify the production rate functions $\mathcal{F}_i$, the discretization may be unchanged and we keep the correspondence between the discrete and continuous system. However, instead of a non-deterministic state transition we now have a natural probability distribution on all neighbours of any given discrete state. This leads to a Markov process that has the same state space as the discrete kinetic logic model. The advantage of the SLN model is that it introduces the noise factor at the continuous level which is more natural than doing so at the discrete level.

2.2 Fast SLN Reconstruction Algorithm

In the previous work [29] we proposed to reconstruct the whole SLN model from discretized time series. This means, that we had to estimate the parameters of the Markov chain with a state space that is exponential in the size of the reconstructed network. For small networks this can be done with very good results using a modified Baum-Welch algorithm [2]. Unfortunately even for networks containing 10 nodes it is impractical both because of the computational complexity of this task and because of the fact that the estimation of so many parameters would require enormous amounts of experimental data.

However, if we are interested only in reconstructing the topology of the SLN network, we can avoid the exhaustive computation. We should recall the procedure for retrieving the topology of the network, given the parameters of the Markov chain from our previous paper [29]. It is based on the fact that the existence of a non-trivial dependency between two genes follows from the presence of the so-called *witness* of regulation. We say that a pair of states $\langle X, X' \rangle$ is a witness of regulation of gene i-th by j-th if these states differ only by the concentration of j-th gene and the ratios of probabilities of raising and decreasing the concentration of gene i-th are different for them (see Figure 1). It is important to note that, assuming the observed data is representative for the true distribution, considering pairs of states is sufficient for reconstruction of all regulatory interactions.

The key observation behind the fast algorithm is that we can look at the observed frequencies instead of transition probabilities in the Markov chain. We can treat the observed state transitions as samples from the state space. Testing if a pair of observed states is a witness of regulation may be now solved by statistical test for the equality of proportions. We should take into account that the sample size may be small so we follow the suggestions of D'Agostino et al. [6]

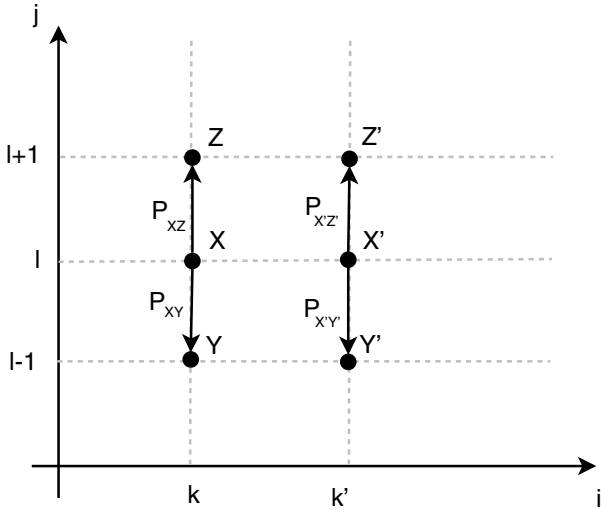


Fig. 1. The projection of the discrete state space on the two dimensions corresponding to genes i and j. The pair of states X an X' witnesses the regulation dependency $i \to j$ if $P_{XZ}/P_{XY} \neq P_{X'Z'}/P_{X'Y'}$ and $X(m) = X'(m)$ for $m \neq i$.

to use of Student t-test in such cases. The only drawback of this method is that we are unable to draw conclusions from unobserved states, so we may recover less interactions than with the previous approach.

The algorithm now is very simple iteration over all pairs of states that can possibly be witnesses of regulation:

Calculate the observed frequencies of state changes
foreach observed state X
  foreach state X' differing from X only by i-th gene
    foreach gene j
      test the hypothesis that $\langle X, X' \rangle$ witness $i \to j$

The complexity of the algorithm is $\mathcal{O}(D \cdot n^2)$, where D denotes the number of observed states and n is the number of nodes in the network.

2.3 Inferring the Type of Regulation

Once we have the topology of regulatory dependencies it is interesting to see if we can predict the type of the dependencies. In general, we are interested in discerning between expression activators and repressors. To this end we may exploit the fact, that for each pair of states $\langle X, X' \rangle$ which witness the regulation $i \to j$ we can judge if the production rate of j grows with increase of i (activation)

or with decrease of i (repression). We assign a type to each edge based on the type suggested by majority of all witnesses for it. In case of a tie we do not assign any type.

In Figures 2-5, the type of dependencies in regulatory networks is traditionally represented by different kinds of arrows: pointed for activators and blunt for repressors.

3 Methods

3.1 Generating Artificial Expression Data

For the first artificial network (see Fig. 2) we have constructed a system of stochastic differential equations consistent with the assumptions of the SLN model. Then, we generated three time series corresponding to each of the cycles in the networks. The data was then discretized into binary values by subtracting from the expression of each gene the mean value of its expression. Then we have chosen all time points where the discrete state was changed which gave altogether 96 pairs of consecutive states.

In the case of the data from the Boolean model by Faure [11], we have selected all the discrete states comprising the cell cycle and selected all state transitions between them that are possible under the assumed model. This gave us 24 pairs of consecutive states, however the state of three genes (CYCD, RB and CKI) remained unchanged under this condition so we have added 3 additional state transitions possible under the assumed model that correspond to the perturbations of these genes.

In the case of the kinetic model of Novak [18], we have used the model files provided by the authors on their website [30] and used the Xpp-aut software [10] to generate the trajectory of the system undergoing three cell cycles. Then we have sampled 150 equally distant time-points and discretized them into binary values using the same procedure as in the first case.

3.2 Microarray Data Material

The dataset published by Whitfield [28] contains 5 time-series containing altogether 106 pairs of consecutive experiments. Unfortunately the expression of the genes corresponding to Cyclin A1 (CYCA) and Cadherin 1 (CDH1) was not measured across all experiments, so they were excluded from the analysis.

3.3 Measuring Quality of Reconstruction

In this work we always assume that we know the correct topology of the network so in order to measure the quality of a reconstruction we need to compare the reconstruction with the original network. This is a reasonable assumption, since our main goal is the verification of the proposed method performance on the data coming from a known source. In general, the ultimate check for the quality of reconstruction is experimental verification of reported interactions, however this is definitely out of scope of this work.

We represent all networks as directed graphs with labels on the edges. To measure the statistical significance of a reconstruction of the network consisting of n nodes correctly assigning k labels to the edges we calculate the probability of obtaining not less than k successes in n Bernoulli trials:

$$\mathcal{P}(n,k) = \sum_{i=k}^{n} \binom{n}{k} p^i (1-p)^{n-i}$$

The probability of success was set to $1/2$ in case of predicting the presence of directed edges, and to $1/3$ if we allow for dependencies of two types (activators and repressors). In Tables 1-4, we report p-values for the model with $p = 1/3$.

The sensitivity and specificity are traditionally defined in the following way:

$$Sens = \frac{TP}{TP + FN},$$
$$Spec = \frac{TN}{TN + FP},$$

where TP, TN, FP and FN represent true positives, true negatives, false positives and false negatives, respectively. The values of sensitivity and specificity presented in Tables 1-4, are calculated for the case of predicting interactions regardless of their type.

3.4 Software Implementation

The algorithm for SLN reconstruction was implemented in Python programming language and is available under the GNU General Public license. It can be downloaded from the supplementary website [32]. We have also implemented a web server allowing for running our method using a web interface [31].

4 Experiments

We have made several experiments in order to test the accuracy of our method. All our tests are made in the same manner and consist of a model with known (or at least postulated) topology, and a discretized expression time-series obtained from this system. For all datasets, we perform the reconstruction using our method and using Dynamic Bayesian Networks (both with MDL and BDE scores) on the same data for comparison. In order to measure the quality of reconstruction for all evaluated methods we treat the reconstructed networks as predictions, where for each ordered pair of genes, the method independently predicts the presence and type of dependency. In this setting we can use standard measures for the quality of prediction:

- number of predicted dependencies
- Number of correctly predicted dependencies (with and without type),

– p-value of such prediction quality using binomial distribution as the null-model (we report p-value for the model with $p = 1/3$ suitable for predicting labelled edges),
– sensitivity and specificity of this prediction ().

All reconstructed networks are presented in Figures 2-5, with correctly reconstructed edges presented as solid lines and incorrectly predicted as dotted lines, the color versions of the images are available from the supplementary website (http://bioputer.mimuw.edu.pl/papers/bw/cmsb07).

4.1 Artificial Networks

Since SLN model was tailored for reconstruction of homeostatic systems, we have tested our method on synthetic data obtained by simulation of very simplistic homeostatic loops proposed by Thomas and D'ari [27]. According to the seminal work of [26], the logical network is capable of homeostatic behaviour only if it contains a cycle with an odd number of negative regulatory dependencies. The simplest possible network for benchmarking of our method would consist of a single negative cycle. In the previous paper we were able to analyze such networks of size 2, 3 and 4 but could not scale it to larger networks. We have verified that, the algorithm presented here produces exactly the same results (not shown) for these small networks as the previous method [29]. Since this method is capable of dealing with much larger networks, we have constructed a simplistic network consisting of three independent feedback loops containing four genes each, presented in Fig. 2 (a). The reconstructed networks are presented graphically in Figure 2 and the performance of all methods on this dataset is given in the corresponding Table 1.

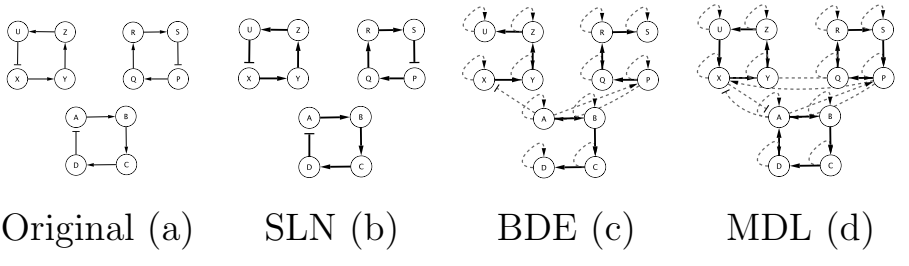


Fig. 2. Comparison of the networks reconstructed from the synthetic data using different methods to the one postulated by the Boolean model. Solid lines represent correctly reconstructed edges, dotted lines are false positives.

4.2 Discrete Boolean Network Model of Mammalian Cell Cycle

Recently Faure et al. [11] published a study summarizing current knowledge of the regulatory dependencies among key mammalian cell cycle regulators. They present a topology of the network as well as the regulatory functions, so the

Table 1. Results of the reconstruction from artificial data from simple Boolean network consisting of 3 4-gene loops (12 vertices, 12 edges)

method	Edges	Correct(w/type)	p-value	Sens.	Spec.
SLN	12	12 (12)	$4.4 \cdot 10^{-70}$	1.00	1.00
DBN(BDE)	24	9 (9)	$6.3 \cdot 10^{-42}$	0.75	0.89
DBN(MDL)	30	12 (9)	$1.7 \cdot 10^{-35}$	1.00	0.86

types of all relationships are known. The mathematical model they choose for representation of the network is the kinetic logic approach with slightly modified state transition. Instead of fully non-deterministic and asynchronous state changes, the authors choose a mixed model, where some genes act synchronously whereas others change their state asynchronously. The model they present shows cyclic behaviour under natural conditions during cell cycle and agrees with the knowledge on the behaviour of some mutants. We have analyzed the dataset representing the natural cell cycle in this model. The result are presented in Figure 3 and Table 2.

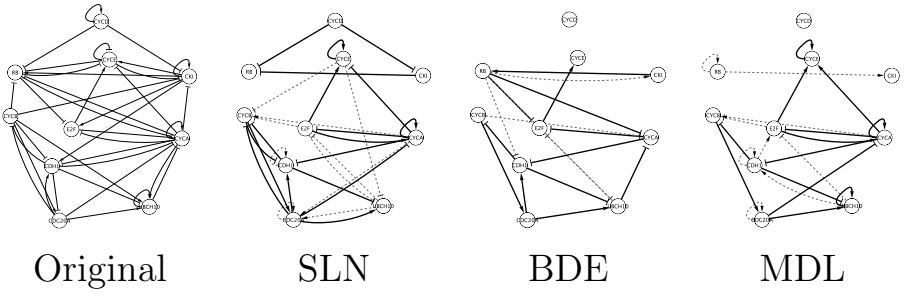


Fig. 3. Comparison of the networks reconstructed from the synthetic data using different methods to the one postulated by the Boolean model. Solid lines represent correctly reconstructed edges, dotted lines are false positives.

Table 2. Results of the reconstruction from artificial data from Boolean network published in [11] (10 vertices, 38 edges)

method	Edges	Correct(w/type)	p-value	Sens.	Spec.
SLN	28	18 (16)	$1.6 \cdot 10^{-11}$	0.47	0.84
DBN(BDE)	16	8 (7)	$2.9 \cdot 10^{-8}$	0.21	0.87
DBN(MDL)	22	13 (11)	$2.8 \cdot 10^{-9}$	0.34	0.84

4.3 Continuous Kinetic Model of Mammalian Cell Cycle

An earlier study of Novak and Tyson [18] proposes a more detailed kinetic model of the mammalian cell cycle. This model aims to provide more quantitative data on the regulation of the cell cycle. The set of genes is slightly different (the

UBCH10 and RB proteins are included only in the model by Faure [11]), but for all other gene products the kinetic equations describing the system dynamics are provided. The authors provide the model in the form of ordinary differential equations allowing for generating trajectories of the system, so we have used one such trajectory to test our method. The expression profiles of all genes were generated in 150 points over three cell cycles and discretized into binary values. The results are summarized in Table 3 and the resulting networks are presented in Figure 4.

In order to provide the correct topology of the regulatory dependencies we assume the existence a regulatory dependence of X on Y if a concentration of a protein X has a non-zero effect on the derivative of the concentration of protein Y. Although this is the most straightforward way of defining topology it should be noted, that it ignores the impact of indirect dependencies, especially if the intermediate entity is not a single protein (e.g. a protein complex) and thus its concentration is not taken into account in the analyzed dataset.

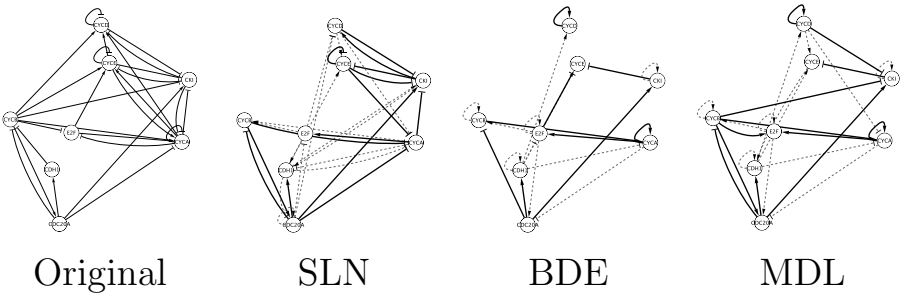


Fig. 4. Comparison of the networks reconstructed from the synthetic data using different methods to the one postulated by the kinetic model. Solid lines represent correctly reconstructed edges, dotted lines are false positives.

Table 3. Results of the reconstruction from kinetic model data published in [18] (8 vertices, 27 edges)

method	Edges	Correct(w/type)	p-value	Sens.	Spec.
SLN	29	14 (10)	$4.7 \cdot 10^{-2}$	0.51	0.59
DBN(BDE)	20	9 (5)	$7.8 \cdot 10^{-2}$	0.33	0.70
DBN(MDL)	25	12 (8)	$4.7 \cdot 10^{-2}$	0.44	0.65

4.4 Reconstruction of Mammalian Cell Cycle Network from Microarray Data

A final, and most important test of our method is the evaluation of its performance on the experimental Microarray data. For mammalian cell cycle there are two publicly available published datasets: the study of cell cycle in human fibroblasts by Cho et al. [5] and the work of Whitfield et al. [28] based on the

HeLa cancer cell line. Interestingly enough, the analysis of data from the first study has given results of no statistical significance at all (data presented on supplementary website [32]) for our method, as well as for the Dynamic Bayesian approach. This is not surprising, since Shedden and Cooper [23] have analysed this very dataset and concluded that the level of noise in the data is so high that it is not feasible to recover any cyclic behaviour of genes at all. The authors also suggest that the reason behind this problem is the improper synchronization of the cells.

Unfortunately, the second dataset is also carrying a bias, since it was obtained from cancer cells. Nonetheless, both our method and Dynamic Bayesian approach were able to correctly recover approximately one third of the true edges. The results are presented in Table 4 and Figure 5.

Table 4. Results of the reconstruction from experimental microarray data published in [28] (8 vertices, 20 edges)

method	Edges	Correct(w/type)	p-value	Sens.	Spec.
SLN	19	7 (5)	$2.9 \cdot 10^{-4}$	0.35	0.73
DBN(BDE)	16	6 (3)	$7.0 \cdot 10^{-4}$	0.30	0.73
DBN(MDL)	39	12 (6)	$8.9 \cdot 10^{-1}$	0.60	0.39

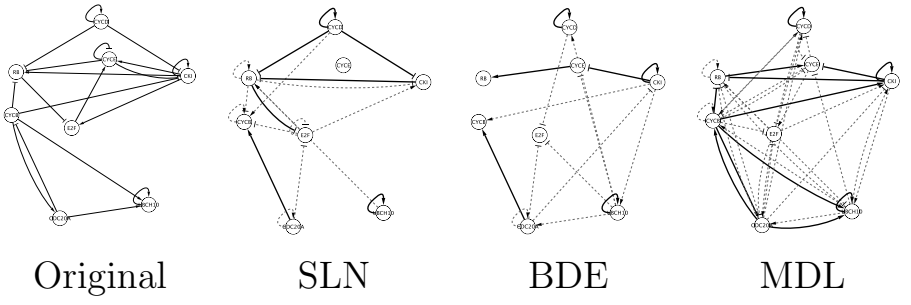


Fig. 5. Comparison of the networks reconstructed from the experimental microarray data using different methods to the topology postulated by the most recent Boolean model. Solid lines represent correctly reconstructed edges, dotted lines are false positives.

5 Discussion

The problem of network topology reconstruction from high-throughput expression data is very important to help us understand the complex mechanisms behind biological processes. The results of such analyses should be regarded as an input to further experimental validation rather than as a complete solution. We see our method as a potentially helpful tool to indicate plausible regulatory dependencies among any group of genes interesting from biological point of view. Before such a tool may be put to use, one has to validate its performance on realistic data.

In this paper, we take advantage of the presence of a well studied system such as the mammalian cell cycle, for which we have the necessary experimental data and corresponding formal models of the regulatory interactions. Using all this information we can properly evaluate the performance of our proposed method also in comparison with a different approach through Dynamic Bayesian Networks.

The results of the experiments performed on the simplest data from artificial network show that the presented faster algorithm yields exactly the expected results if there is enough data generated from assumed model.

In the second case, where data was generated from a very similar Boolean model, our method was still able to provide highly significant results reconstructing half of the true interactions with a relatively small number of wrong predictions (84% sensitivity). It is even more interesting if we take into account the fact that the second prediction was based on significantly less data-points than the first one. The performance of the DBN approach is comparable in terms of statistical significance and specificity (approx. 85%), but it shows significantly lower sensitivity.

The third dataset obtained with a kinetic model of the cell cycle surprisingly seems to be even more difficult for both methods than the microarray data. The statistical significance of the reconstructed networks ($0.1 > p > 0.01$) is questionable and the sensitivity and specificity are not satisfactory.

The results of our method for the real microarray data is considerably higher that in the previous case. Even though we have reconstructed little over one third of the edges in the correct network (and only 5/19 with correctly assigned type of dependency) the specificity of our prediction is much higher than the one obtained for previous dataset. At the same time it is quite surprising, that the performance of DBN approach on this dataset is highly dependent on the choice of the scoring function: while BDE score performs comparably with our method, MDL returns almost three times more interactions which makes its result close to random.

In summary, our method seems to perform comparably or better than Dynamic Bayesian Networks on all considered datasets. It should be noted, that in all cases (ruling out the statistically insignificant MDL prediction for microarray data) our method correctly reconstructed more edges than DBN method, also the precision ratio of correctly predicted edges to all predicted edges used by Shi et al. [24] favours our method above the DBN approach. It is also clear that both methods (SLN and DBN) are able to arrive at much higher recall rates (the fraction of all interactions that were correctly predicted used by Shi [24]) that the methods suited for use on the genome scale [24,33].

It should be also noted, that due to the fast algorithm for SLN reconstruction we were able to compute all the results of the presented experiments in times well under 10 seconds on a standard PC which is considerably faster than the reconstruction of DBN models (especially in the case of BDE score) which for the same data took up to a minute on the same hardware.

6 Conclusion

In this paper, we present a novel approach to the problem of regulatory network reconstruction from microarray data. Our algorithm is polynomial in the size of the data so it can be applied to datasets of realistic size. An important part of this work is the analysis of the performance of the method in comparison with Dynamic Bayesian Network approach on different datasets, both synthetic and experimental. The results of the experiments show that both methods can be considered a valuable tool for the task of finding regulatory dependencies from expression data and that our method yields consistently better results than DBNs.

Acknowledgements

We would like to thank Dr. Norbert Dojer for help with using the Dynamic Bayesian Network method.

This work was partially supported by the Polish Ministry of Science grants No 3 T11F 021 28 and PBZ-MNiI-2/1/2005. The computational resources were provided by CoE BioExploratorium project: WKP_1/1.4.3/1/2004/44/44/115/2005. BW received financial support from Foundation for Polish Science.

References

1. Akutsu, T., Miyano, S., Kuhara, S.: Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In: Pacific Symposia on Biocomputing, pp. 17–28 (1999)
2. Baum, L.E., Peterie, T., Souled, G., Weiss, N.: A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Statist. 41(1), 164–171 (1970)
3. Chaouiya, C., de Jong, H., Thieffry, D.: Dynamical modeling of biological regulatory networks. Biosystems 84(2), 77–80 (2006)
4. Chen, K.-C., Wang, T.-Y., Tseng, H.-H., Huang, C.-Y., Kao, C.-Y.: A stochastic differential equation model for quantifying transcriptional regulatory network in Saccharomyces cerevisiae. Bioinformatics Evaluation Studies 21(12), 2883–2890 (2005)
5. Cho, R.J., Huang, M., Campbell, M.J., Dong, H., Steinmetz, L., Sapinoso, L., Hampton, G., Elledge, S.J., Davis, R.W., Lockhart, D.J.: Transcriptional regulation and function during the human cell cycle. Nat. Genet. 27(1), 48–54 (2001)
6. D'Agostino, R.B., Chase, W., Belanger, A.: The appropriateness of some common proceduresfor testing the equality of two independent binomial populations. The American Statistician 42(3), 198–202 (1988)
7. Eric, H.: Davidson, Rast, Oliveri, Ransick, Calestani, Yuh, Minokawa, Amore, Hinman, Arenas-Mena, Otim, Brown, Livi, Lee, Revilla, Rust, Pan, Schilstra, Clarke, Arnone, Rowen, Cameron, McClay, Hood, Bolouri, and Davidson EH. A genomic regulatory network for development. Science (2002)
8. Dojer, N.: Learning Bayesian networks does not have to be NP-hard. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 305–314. Springer, Heidelberg (2006)

9. Dojer, N., Gambin, A., Wilczyński, B., Tiuryn, J.: Applying dynamic Bayesian networks to perturbed gene expression data. BMC Bioinformatics 7, 249 (2006)
10. Ermentrout, B.: Simulating, Analyzing, and Animating Dynamical Systems: A Guide to Xppaut for Researchers and Students. In: Society for Industrial and Applied Mathematics, 1st edn. (2002)
11. Faure, A., Naldi, A., Chaouiya, C., Thieffry, D.: Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. Bioinformatics 22(14), 124–131 (2006)
12. Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using bayesian networks to analyze expression data. Journal of Computational Biology 7, 601–620 (2000)
13. Husmeier, D.: Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. Bioinformatics 19(17), 2271–2282 (2003)
14. Imoto, S., Goto, T., Miyano, S.: Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. In: Pacific Symposia on Biocomputing, pp. 175–186 (2002)
15. Liang, S., Fuhrman, S., Somogyi, R.: Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In: Pacific Symposia on Biocomputing, pp. 18–29 (1998)
16. Mendoza, L., Thieffry, D., Alvarez-Buylla, E.R.: Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analyssis. Journal of Theoretical Biology (1999)
17. Murphy, K., Mian, S.: Modelling gene expression data using dynamic Bayesian networks. University of California, Berkeley (1999)
18. Novak, B., Tyson, J.: A model for restriction point control of the mammalian cell cycle. J. Theor. Biol. 230(4), 563–579 (2004)
19. Ott, S., Imoto, S., Miyano, S.: Finding optimal models for gene networks. In: Proc. of Pacific Symposia in Biocomputing (2004)
20. Parkinson, H., Sarkans, U., Shojatalab, M., Abeygunawardena, N., Contrino, S., Coulson, R., Farne, A., Lara, G.G., Holloway, E., Kapushesky, M., Lilja, P., Mukherjee, G., Oezcimen, A., Rayner, T., Rocca-Serra, P., Sharma, A., Sansone, S., Brazma, A.: ArrayExpress–a public repository for microarray gene expression data at the EBI. Nucleic Acids Res. 33, 553–555 (2005)
21. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan-Kauffman, San Francisco, CA (1988)
22. Sanchez, L., Thieffry, D.: Segmenting the fly embryo: a logical analysis of the pair-rule cross-regulatory module. J. Theor. Biol. 224(4), 517–537 (2003)
23. Shedden, K., Cooper, S.: Analysis of cell-cycle-specific gene expression in human cells as determined by microarrays and double-thymidine block synchronization. Proc. Natl. Acad. Sci. U S A 99(7), 4379–4384 (2002)
24. Shi, Y., Mitchell, T., Bar-Joseph, Z.: Inferring Pairwise Regulatory Relationships from Multiple Time Series Datasets. Bioinformatics, JOURNAL ARTICLE (January 2007)
25. Shmulevich, I., Dougherty, E., Kim, S., Zhang, W.: Probabilistic Boolean Networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics 18(2), 261–274 (2002)
26. Thomas, R.: Boolean formalization of genetic control circuits. Journal of Theoretical Biology 42, 563 (1973)
27. Thomas, R., D'Ari, R.: Biological Feedback. CRC Press, Boca Raton, USA (1990)

28. Whitfield, M.L., Sherlock, G., Saldanha, A.J., Murray, J.I., Ball, C.A., Alexander, K.E., Matese, J.C., Perou, C.M., Hurt, M.M., Brown, P.O., Botstein, D.: Identification of genes periodically expressed in the human cell cycle and their expression in tumors. Mol. Biol. Cell. 13(6), 1977–2000 (2002)
29. Wilczyński, B., Tiuryn, J.: Regulatory network reconstruction using Stochastic Logical Networks. In: Priami, C. (ed.) CMSB 2006. LNCS (LNBI), vol. 4210, pp. 142–154. Springer, Heidelberg (2006), http://dx.doi.org/10.1007/11885191_10
30. Novak and tyson cell cycle model webpage: http://mpf.biol.vt.edu/research/generic_model
31. Sln webserver: http://bioputer.mimuw.edu.pl/BIAS/SLN/
32. Mammalian cell cycle reconstruction – supplementary webpage: http://bioputer.mimuw.edu.pl/papers/bw/cmsb07/
33. Zhao, W., Serpedin, E., Dougherty, E.: Inferring gene regulatory networks from time series data using the minimum description length principle. Bioinformatics 22(17), 2129–2135 (2006)

An Automated Translation from a Narrative Language for Biological Modelling into Process Algebra

Maria Luisa Guerriero[1], John K. Heath[2], and Corrado Priami[1,3]

[1] Dept. of Information and Communication Technology, University of Trento, Italy
[2] CRUK Growth Factor Group, School of Biosciences, University of Birmingham, UK
[3] The Microsoft Research - University of Trento
Centre for Computational and Systems Biology, Italy
{guerrier,priami}@dit.unitn.it, j.k.heath@bham.ac.uk

Abstract. The aim of this work is twofold. First, we propose an high level textual modelling language, which is meant to be biologically intuitive and hence easily usable by life scientists in modelling intra-cellular systems. Secondly, we provide an automatic translation of the proposed language into Beta-binders, a bio-inspired process calculus, which allows life scientists to formally analyse and simulate their models. We use the Gp130 signalling pathway as a case study.

1 Introduction and Motivations

Process calculi, originally developed for modelling mobile communicating systems [1], have proved to be both useful and powerful tools for modelling biological signalling pathways [2,3,4,5,6]. The need for such modelling approaches has arisen from the complexity of these processes, which is not easily analysed by biological intuition [7]. A key challenge in the application of process calculi to biological problems is the specification of models. The issue poses several practical and computational problems. First, it is unlikely that most practising biologists would be motivated to formulate their ideas in a non-intuitive language devised for computational execution. Second, the process of specifying the model can identify areas of ignorance or uncertainty in biological understanding which may not be apparent to a computer scientist taking models from the literature. The biological literature frequently articulates biological models in the form of informal diagrams or employs various formal graphical notations (e.g. Kohn molecular interaction maps [8] and Kitano diagrams [9]), but these can be confusing or ambiguous and particularly may conceal or abstract biological knowledge that may be useful for the computational modeller. It is also clear that a two-dimensional static representation may not adequately represent all pertinent features of the dynamic evolution of a temporal and spatially directed process. Indeed these ways of describing a biological process in reality involve the biologist translating a narrative of events into a graphical format.

Prompted by these considerations we here present an approach to biological model specification which is based on a narrative style language. In developing

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 136–151, 2007.

this language there were three desiderata. First, the language should be biologically intuitive using formalisms and syntax familiar to biologists. Second, the language should exploit the specific advantages of a process calculus based approach to modelling highlighting, for example, the roles of concurrency, dependency and spatial confinement. Thirdly, the language should encourage the biologists to critically examine their understanding in the process of model specification highlighting points of uncertainty.

In the proposed framework, biologists can specify their model by providing a textual description of the system, containing the list of compartments, the list of entities (i.e. proteins) composing the model, the list of reactions with their rate parameters, and a narrative describing the evolution of the system.

A translation into process calculus (namely, Beta-binders [4]) has been developed and a few models have been specified into the proposed language and translated into Beta-binders. This allowed us to run the models by using the BetaWB simulator [10].

The front-end of the translation is designed to be independent of the language, so that when implementing a translation to other modelling languages it could be reused. The main aim of the current work, in fact, is to define an high level language which hides the implementation details to the modeller, and is generic so that it could be translated into other kinds of formalisms different from Beta-binders (e.g. other process algebras, SBML [11], and possibly differential equations). This allows us to have a common ground in which models could be specified, and it also allows modellers to choose the output language that better fits their aims. In fact it is difficult to find one formal language which is appropriate to describe all kinds of systems, or that could be used to study all aspects of the systems. In addition, starting with the same input model, the comparison of different formalisms would be much easier.

We use as a case study the Gp130 signalling pathway [12]. This was chosen as an example of signalling concurrency which has yet to be explicitly articulated in a process calculus model.

In Sect. 2 the modelling language is described, using the Gp130 pathway as a modelling example. In Sect. 3 the translation algorithm is described, and the generated Gp130 pathway Beta-binders model is shown. Finally, we draw some conclusions on the pros of our approach and on issues that need to be tackled.

2 The Narrative Language

In this section we present the modelling language. The structure of models in the proposed language is inspired by SBML [11], one of the most well known description languages for biology. The main differences between the two languages are the following ones. First, SBML species have only one possible state, while in our language they can have multiple states (this choice was made bearing in mind that our target language, process algebra, is a multistate one). Second, SBML is very abstract and it does not represent details about species and reactions, while the proposed language can easily represent them. Finally, the description

of system evolution in SBML is given in terms of reactions with reactants and products, while in our language it is given in terms of a narrative of events.

A model in our language is composed of four sections:

- the description of the biological compartments in which the involved entities can be located during the evolution of the system;
- the description of the entities composing the system;
- the description of the occurring reactions;
- the narrative description of the evolution of the system, i.e. the list of the occurring events.

A *compartment* is identified by an integer number; moreover, its name, size, and number of spatial dimensions can be specified. Compartments could represent cellular or sub-cellular compartments, but also abstract locations.

A *component* (i.e. a protein) is identified by its name, and it can be seen as a list of interaction sites. Each site is defined by a name and a state (e.g. phosphorylated, bound, active, etc.). This choice reflects the fact that each basic event in the evolution of intra-cellular systems is the modification of one interaction site for each protein involved in the reaction. However, since interaction sites are not always known, states can also be associated to the protein itself, to represent modifications involving generic interaction sites. If the position of the protein is relevant, the compartments in which it can be located during the system evolution can be specified. Finally, the initial quantity/concentration of the component should be set.

A *reaction* is identified by an integer number; its type (e.g. phosphorylation, binding, etc.) and the reaction rate parameter should also be specified.

A reliability value can be associated to each numerical value (e.g. rate parameters and initial quantities); it is a percentage value that can be used to distinguish between values that are certain because obtained from wet experiments, and others which are the result of not verified assumptions. Modellers can take this information into account during the important step of model refining by means of parameter space search and sensitivity analysis.

Finally, the evolution of the system is described by means of a *narrative of events*. This narrative is a sequence of basic events, each of which is a textual description of a reaction involving at most two components. Events can be grouped into processes.

An *event* is identified by an integer number; in addition to the textual semi-formal description of the event, the identifier of the reaction associated with the event should be specified. The description of the event is a string of the form ***if*** *condition* ***then*** *event_descr*, with the conditional part being optional. A *condition* is a string of the form *component* ***is*** *state*, *component.site* ***is*** *state*, *component* ***is in*** *compartment*, etc. Multiple conditions can be specified by separating them with the keyword ***and***. *Event_descr* is a string of the form *component reaction* for monomolecular events, or *component reaction component* for bimolecular events, where *reaction* can be for example **phosphorylates**, **dephosphorylates**, **binds**, **activates**. As mentioned before, we assume that each

event involves an interaction or modification of one site for each involved protein. If the exact site is known, it can be specified with a clause of the form ***on*** *sites*; otherwise, it is assumed that one of the component's defined states is involved. Hence, conceptually, each step involves a single site; however, a list of sites (separated by semicolons) can be specified as a shortcut for simultaneous steps involving different sites (e.g. simultaneous phosphorylation of two sites).

In real life, two events occurring in a system of interacting entities can be either concurrent (independent events, e.g. events involving different proteins), or sequential (one event can occur only after the other one has occurred, e.g. a phosphorylation of a site of a protein allowed only after it is bound to another protein), or alternative (if one event occurs, the other one cannot occur, e.g. binding of competing ligands to a receptor).

Consequently, in our language it is necessary to distinguish between concurrent, sequential, and alternative events. If a reaction event is alternative to another one, the identifier of the alternative event should be specified. Conditions should be used to enforce the ordering of sequential events. Events that are not explicitly declared either alternative or sequential, are considered independent and are treated as concurrent events.

The grammar of the language follows. Some attributes are left as optional and are not actually used in the translation to Beta-binders, because they refer to details that cannot be easily handled in Beta-binders, or about which we are not interested at the moment; they have been added to allow them to be used in an hypothetical translation to other languages that might handle those details.

$$
\begin{array}{lll}
\langle model\rangle & ::= & \langle comparts\_decl\rangle\langle compons\_decl\rangle\langle reacts\_decl\rangle\langle procs\_decl\rangle \\
\langle comparts\_decl\rangle & ::= & \text{Compartments } \langle comparts\_list\rangle \\
\langle compons\_decl\rangle & ::= & \text{Components } \langle compons\_list\rangle \\
\langle reacts\_decl\rangle & ::= & \text{Reactions } \langle reacts\_list\rangle \\
\langle procs\_decl\rangle & ::= & \text{Narrative } \langle procs\_list\rangle \\
 & & \\
\langle comparts\_list\rangle & ::= & \langle compartment\rangle \\
 & | & \langle compartment\rangle\langle comparts\_list\rangle \\
\langle compons\_list\rangle & ::= & \langle component\rangle \\
 & | & \langle component\rangle\langle compons\_list\rangle \\
\langle reacts\_list\rangle & ::= & \langle reaction\rangle \\
 & | & \langle reaction\rangle\langle reacts\_list\rangle \\
\langle procs\_list\rangle & ::= & \langle proc\rangle \\
 & | & \langle proc\rangle\langle procs\_list\rangle \\
 & & \\
\langle compartment\rangle & ::= & (\langle id\rangle, \langle compart\_name\rangle, \langle opt\_size\rangle, \langle opt\_unit\rangle, \langle opt\_dim\rangle) \\
\langle component\rangle & ::= & (\langle name\rangle, \langle opt\_inform\_descr\rangle, \langle opt\_sites\_def\rangle, \\
 & & \langle opt\_states\_def\rangle, \langle opt\_comparts\_def\rangle, \langle initial\_quantity\rangle) \\
\langle reaction\rangle & ::= & (\langle id\rangle, \langle react\_type\rangle, \langle rate\_const\rangle) \\
\langle proc\rangle & ::= & \text{Process } \langle opt\_inform\_descr\rangle\langle events\_list\rangle \\
\langle events\_list\rangle & ::= & \langle event\rangle \\
 & | & \langle event\rangle\langle events\_list\rangle \\
\langle event\rangle & ::= & (\langle id\rangle, \langle form\_descr\rangle, \langle react\_id\rangle, \langle opt\_altern\_event\rangle)
\end{array}
$$

$\langle opt\_sites\_def \rangle$ ::=
| $\langle sites\_def \rangle$
$\langle sites\_def \rangle$ ::= $\langle site\_def \rangle$
| $\langle site\_def \rangle ; \langle sites\_def \rangle$
$\langle site\_def \rangle$ ::= $\langle name \rangle : \langle state\_name \rangle : \langle is\_active \rangle$

$\langle opt\_states\_def \rangle$ ::=
| $\langle states\_def \rangle$
$\langle states\_def \rangle$ ::= $\langle state\_def \rangle$
| $\langle state\_def \rangle ; \langle states\_def \rangle$
$\langle state\_def \rangle$ ::= $\langle state\_name \rangle : \langle is\_active \rangle$

$\langle opt\_comparts\_def \rangle$ ::=
| $\langle comparts\_def \rangle$
$\langle comparts\_def \rangle$ ::= $\langle compart\_def \rangle$
| $\langle compart\_def \rangle ; \langle comparts\_def \rangle$
$\langle compart\_def \rangle$ ::= $\langle id \rangle : \langle is\_active \rangle$

$\langle initial\_quantity \rangle$ ::= $(\langle quantity \rangle, \langle opt\_reliability \rangle)$
$\langle rate\_const \rangle$ ::= $(\langle rate \rangle, \langle opt\_unit \rangle, \langle opt\_reliability \rangle)$

$\langle form\_descr \rangle$ ::= $\langle event\_descr \rangle$
| if $\langle conds \rangle$ then $\langle event\_descr \rangle$

$\langle conds \rangle$ ::= $\langle cond \rangle$
| $\langle cond \rangle$ and $\langle conds \rangle$
$\langle cond \rangle$ ::= $\langle names \rangle$ is $\langle state\_name \rangle$
| $\langle names \rangle$ is not $\langle state\_name \rangle$
| $\langle names \rangle$ is in $\langle id \rangle$
| $\langle names \rangle$ is not in $\langle id \rangle$
$\langle names \rangle$ ::= $\langle name \rangle$
| $\langle name \rangle . \langle name \rangle$
| $\langle name \rangle ; \langle names \rangle$
| $\langle name \rangle . \langle name \rangle ; \langle names \rangle$
$\langle sites \rangle$ ::= $\langle name \rangle$
| $\langle name \rangle ; \langle sites \rangle$

$\langle event\_descr \rangle$ ::= $\langle complex\_name \rangle \langle bimol\_react \rangle \langle complex\_name \rangle$ on $\langle sites \rangle$
| $\langle complex\_name \rangle \langle bimol\_react \rangle \langle complex\_name \rangle$
| $\langle complex\_name \rangle \langle monomol\_react \rangle$ on $\langle sites \rangle$
| $\langle complex\_name \rangle \langle monomol\_react \rangle$
| $\langle complex\_name \rangle$ relocates to $\langle id \rangle$
| $\langle complex\_name \rangle$ degrades
| $\langle complex\_name \rangle$ degrades $\langle complex\_name \rangle$
| $\langle complex\_name \rangle$ synthesises $\langle complex\_name \rangle$
| $\langle complex\_name \rangle$ homodimerises
| $\langle complex\_name \rangle$ dehomodimerises
| $\langle complex\_name \rangle$ dimerises with $\langle complex\_name \rangle$
| $\langle complex\_name \rangle$ dedimerises from $\langle complex\_name \rangle$
$\langle complex\_name \rangle$ ::= $\langle name \rangle$
| $\langle name \rangle : \langle complex\_name \rangle$

⟨*id*⟩ ::= *Int*
⟨*opt_size*⟩ ::=
| *Int*
⟨*opt_unit*⟩ ::=
| *Str*

⟨*opt_dim*⟩ ::=
| *Int*
⟨*name*⟩ ::= *Ide*
⟨*opt_inform_descr*⟩ ::=
| *Str*
⟨*quantity*⟩ ::= *Int* | *Real*
⟨*opt_reliability*⟩ ::=
| *Int*
⟨*rate*⟩ ::= *Int* | *Real* | inf
⟨*react_id*⟩ ::= *Int*
⟨*opt_altern_event*⟩ ::=
| alternative to ⟨*id*⟩
⟨*is_active*⟩ ::= *Bool*

⟨*compart_name*⟩ ::= nucleus | cytosol | exosol
| cellMembrane | nucleusMembrane | *Ide*

⟨*react_type*⟩ ::= phosphorylation | dephosphorylation
| binding | unbinding
| homodimerization | dehomodimerization
| dimerization | dedimerization
| activation | deactivation
| hydrolysis | dehydrolysis
| degradation | synthesis | relocation

⟨*state_name*⟩ ::= phosphorylated | bound | active | hydrolysed | dimer

⟨*bimol_react*⟩ ::= phosphorylates | dephosphorylates | binds | unbinds
| activates | deactivates | hydrolyses | dehydrolyses

⟨*monomol_react*⟩ ::= phosphorylates | dephosphorylates | hydrolyses | dehydrolyses

2.1 Case Study: A Narrative Model of the Gp130 Signalling Pathway

In this section we describe a model of the Gp130 signalling pathway in the proposed language.

The Gp130 pathway is the subject of significant clinical and biological interest, not least due to the key role it plays in human fertility, neuronal repair and haematological development [12]. Robust experimental platforms are available and there is much information on pathway components and behaviour. Various features of this pathway make it an attractive case study for the modelling

approach. First, the Gp130 signalling pathway involves a family of private and public receptors where biological outcomes are dictated by the relative occupation of different receptor combinations. Thus, it exhibits signalling concurrency, which has not been explicitly tackled by computational modelling approaches yet. Moreover, a key feature of the Gp130 system is nuclear/cytoplasmic shuttling of key signalling components whose dynamics have been investigated using imaging techniques; hence, this is an excellent test case for developing spatially confined compartment models which can be tested against high quality datasets.

The model is made of six entities: two ligands (LIF and OSM), three membrane-bound receptors (gp130, LIFR and OSMR) and one effector (STAT3).

Four compartments are involved in the system: the exosol (the extracellular space, where the two ligands are located), the cell membrane (location of the receptors), and the cytosol and the nucleus (compartments between which the effector shuttles). Table 1 lists the compartments with their attributes. The sizes of the compartments are their volumes, except for the cellular membrane whose size, being a 2D compartment, is its surface area.

Table 1. List of compartments (The volumes are calculated based on the average cell radius and ratio between intra-cellular compartments volumes stated in [13])

id	name	size	unit of measure	dimensions
1	exosol	$9.95 \cdot 10^{-12}$	l	3
2	cellMembrane	$12.57 \cdot 10^{-8}$	dm^2	2
3	cytosol	$2.10 \cdot 10^{-12}$	l	3
4	nucleus	$0.25 \cdot 10^{-12}$	l	3

The components representing the involved proteins are listed in Table 2. The definition of each component contains its name, an informal description, the list of sites (each defined by its name, state, and a boolean flag specifying whether it is or not in an active state at system initialisation), the list of protein states (each defined by its name and the active/inactive flag), the list of compartments in which the protein could be located (each defined by its identifier referring to the definition in Table 1, and the active/inactive flag), and finally its initial quantity and the reliability of this numerical value.

Table 3 contains the list of reactions. Each reaction definition contains an identifier, its type, and a rate parameter with its unit of measure and reliability. The rate parameter is the reaction kinetic constant (K_a, K_{off}, etc.). In order to be used in stochastic models (such as Beta-binders), the given kinetic constants need to be translated into stochastic reaction rates. As described in [14], for first order reactions (unbinding, phosphorylation and relocation events) the stochastic rate r is equal to the kinetic rate k; for second order heterologous reactions (binding events) $r = \frac{k}{V \cdot N_A}$, where V is the reaction volume and N_A is Avogadro's number, while for second order homologous reactions (homodimerization events) $r = \frac{2 \cdot k}{V \cdot N_A}$. The reaction volume is the volume in which the reaction occurs. For reactions occurring in a 3D compartment or at the internal side of a 2D

Table 2. List of components (The initial quantities values for the ligands LIF and OSM are calculated based on the known extracellular concentration values, 500pM)

name	descr	site	site_state	site_act	state	state_act	compart	compart_act	initial_quant	reliab
LIF	ligand				bound	false	1	true	3000	100%
OSM	ligand				bound	false	1	true	3000	100%
gp130	receptor	LIF	bound	false	dimer	false	2	true	1000	50%
		OSM	bound	false						
		Y767	phospho	false						
		Y814	phospho	false						
		Y905	phospho	false						
		Y915	phospho	false						
LIFR	receptor	LIF	bound	false	dimer	false	2	true	1000	50%
		OSM	bound	false						
		Y981	phospho	false						
		Y1001	phospho	false						
		Y1028	phospho	false						
OSMR	receptor	OSM	bound	false	dimer	false	2	true	1000	50%
		Y917	phospho	false						
		Y945	phospho	false						
STAT3	effector	Y705	phospho	false	dimer	false	3	true	5000	0%
		gp130	bound	false			4	false		
		LIFR	bound	false						
		OSMR	bound	false						

Table 3. List of reactions

id	type	rate	unit of measure	reliability
1	binding	$8 \cdot 10^5$	$M^{-1}s^{-1}$ (k_a)	50%
2	unbinding	$6 \cdot 10^{-4}$	s^{-1} (k_{off})	50%
3	binding	$8 \cdot 10^5$	$M^{-1}s^{-1}$ (k_a)	50%
4	unbinding	$6 \cdot 10^{-3}$	s^{-1} (k_{off})	50%
5	binding	$8 \cdot 10^5$	$M^{-1}s^{-1}$ (k_a)	50%
6	unbinding	$6 \cdot 10^{-3}$	s^{-1} (k_{off})	50%
7	binding	$8 \cdot 10^5$	$M^{-1}s^{-1}$ (k_a)	50%
8	unbinding	$6 \cdot 10^{-4}$	s^{-1} (k_{off})	50%
9	binding	$8 \cdot 10^5$	$M^{-1}s^{-1}$ (k_a)	50%
10	unbinding	$6 \cdot 10^{-4}$	s^{-1} (k_{off})	50%
11	dimerization	inf	$M^{-1}s^{-1}$	50%
12	phosphorylation	0.2	s^{-1} (k_{cat})	50%
13	binding	10^4	$M^{-1}s^{-1}$ (k_a)	50%
14	dimerization	inf	$M^{-1}s^{-1}$ (k_a)	50%
15	phosphorylation	0.2	s^{-1} (k_{cat})	50%
16	unbinding	10^{-3}	s^{-1} (k_{off})	50%
17	homodimerization	inf	s^{-1}	50%
18	relocation	10	min ($t_{1/2}$)	50%
19	relocation	100	min ($t_{1/2}$)	50%

membrane enclosing a compartment, it is the volume of the compartment. For reactions occurring on a membrane or at its external side, it is the volume of a spherical shell, external to the membrane, of a given radius (e.g. for cell membrane, this radius can be assumed to be half of the cell radius).

Table 4. List of events

id	description	react	alt
	LIF-LIFR binding		
1	if LIFR.LIF is not bound and LIF is not bound then LIF binds LIFR on LIF	1	
2	if LIFR.LIF is bound and LIF is bound then LIF unbinds LIFR on LIF	2	
	LIF-gp130 binding		
3	if gp130.LIF is not bound and LIF is not bound then LIF binds gp130 on LIF	3	
4	if gp130.LIF is bound and LIF is bound then LIF unbinds gp130 on LIF	4	
	OSM-LIFR binding		
5	if LIFR.OSM is not bound and OSM is not bound then OSM binds LIFR on OSM	5	1
6	if LIFR.OSM is bound and OSM is bound then OSM unbinds LIFR on OSM	6	
	OSM-OSMR binding		
7	if OSMR.OSM is not bound and OSM is not bound then OSM binds OSMR on OSM	7	
8	if OSMR.OSM is bound and OSM is bound then OSM unbinds OSMR on OSM	8	
	OSM-gp130 binding		
9	if gp130.OSM is not bound and OSM is not bound then OSM binds gp130 on OSM	9	3
10	if gp130.OSM is bound and OSM is bound then OSM unbinds gp130 on OSM	10	
	LIF pathway		
11	if LIFR.LIF is bound then LIFR dimerises with gp130	11	2
12	if gp130 is dimer then gp130 phosphorylates on Y767;Y814;Y905;Y915	12	
13	if LIFR is dimer then LIFR phosphorylates on Y981;Y1001;Y1028	12	
	OSM pathway		
14	if LIFR.OSM is bound then LIFR dimerises with gp130	11	6
15	if OSMR.OSM is bound then OSMR dimerises with gp130	14	8
16	if gp130 is dimer then gp130 phosphorylates on Y767;Y814;Y905;Y915	12	
17	if OSMR is dimer then OSMR phosphorylates on Y917;Y945	12	
	STAT3 pathway		
18	if gp130.Y767 is phospho and STAT3 is in 3 then gp130 binds STAT3 on gp130	13	
19	if LIFR.Y981 is phospho and STAT3 is in 3 then LIFR binds STAT3 on LIFR	13	
20	if OSMR.Y917 is phospho and STAT3 is in 3 then OSMR binds STAT3 on OSMR	13	
21	if STAT3.gp130 is bound then STAT3 phosphorylates on Y705	15	
22	if STAT3.LIFR is bound then STAT3 phosphorylates on Y705	15	
23	if STAT3.OSMR is bound then STAT3 phosphorylates on Y705	15	
24	if STAT3.gp130 is bound and STAT3.Y705 is phospho then gp130 unbinds STAT3 on gp130	16	
25	if STAT3.LIFR is bound and STAT3.Y705 is phospho then LIFR unbinds STAT3 on LIFR	16	
26	if STAT3.OSMR is bound and STAT3.Y705 is phospho then OSMR unbinds STAT3 on OSMR	16	
27	if STAT3.Y705 is phospho and STAT3.gp130 is not bound and STAT3.LIFR is not bound and STAT3.OSMR is not bound then STAT3 homodimerises	17	
28	if STAT3 is in 3 and STAT3 is dimer then STAT3 relocates to 4	18	
29	if STAT3 is in 4 then STAT3 relocates to 3	19	

Finally, the narrative of events is shown in Table 4. The events are grouped into processes, relative to the binding/unbinding of ligand/receptor pairs, the downstream LIF and OSM pathways, and the downstream STAT3 pathway, which starts with the binding of STAT3 to one of the receptors and ends in its translocation into the nucleus. Each event is described by an identifier, the semiformal description, the identifier of the reaction referring to the definition in Table 3 and the optional identifier of the alternative event.

3 The Translation into Beta-binders

3.1 Beta-binders

Beta-binders [4] is a language, which belongs to the family of the bio-inspired process calculi, strongly inspired by pi-calculus. The main advantage of process

calculi is that, in addition to simulation, they allow for analysing statically the models (e.g. causality, locality, equivalence and reachability analysis). They also allow the modeller to easily execute so called in silico genetics experiments, i.e. to modify some components of the system, and execute simulations to verify the modified system behaviour.

Beta-binders language is quite new, but much work has been done with it in the past few years, and a simulator has also been recently developed [10].

Beta-binders was developed to better adhere to the structure and dynamics of biological systems. By introducing the concept of *affinity*, the calculus relaxes the *key-lock* model of interaction, commonly assumed in classical process calculi, and hence it permits to model more correctly domains and interactions between enzymes and small molecules based on their types and affinities. In Beta-binders, *pi-processes* are encapsulated into *boxes* (also called *bio-processes*) with interaction capabilities, represented by specialised binders (called *beta binders*). Beta binders have the form $\beta(x : \Gamma)$ (active) or $\beta^{\mathrm{h}}(x : \Gamma)$ (hidden) where the name x is the subject of the beta binder and Γ represents the type of x. The main actions that bio-processes can execute are communications ($x(y)$ for input and $\overline{x}\langle y\rangle$ for output) and operations to manipulate the interaction sites of the boxes ($\mathsf{expose}(x, \Gamma)$, $\mathsf{hide}(x)$, $\mathsf{unhide}(x)$ and $\mathsf{chtype}(x, \Gamma)$). These actions can be prefixed by conditions on the visibility and on the type of beta binders. The system is a parallel composition of bio-processes that can be either the deadlock bio-process Nil or the elementary bio-process $\mathbf{B}[\, P\,]$.

The reader is referred to [10] for a detailed description of the syntax and semantics of the language.

3.2 The Translation Algorithm

Beta-binders bio-processes are an intuitive representation of proteins, and hence in the translation into Beta-binders we choose to have one bio-process for each component. Each component interaction site and state is translated into one beta binder on the interface of the respective bio-process, and its active/inactive state is represented by different types of the binder (the initial type is given by the active/inactive flag value); similarly, the location of the component is also translated into a beta binder whose type represents the current location of the component.

The translation is subdivided in two main steps. First, one bio-process is created for each component (Algorithm 1.). Then, the pi-processes representing the translation of events are added to the bio-processes (Algorithm 7.).

As Algorithm 1. describes, the bio-processes are initially empty, and then beta binders are added on their interface.

The names and types of the beta binders representing possible compartments, states and sites are constructed based on their names. As Algorithm 2. shows, the type of the beta binder representing a compartment is the compartment name concatenated with the compartment identifier. In the example, STAT3 is located in compartment 3 (the cytosol), hence a binder $\beta(loc : cytosol\_3)$ is added to $STAT3$ bio-process.

Algorithm 1. ComponentsToBioprocesses

```
1: for all component ∈ Components do          ▷ each component is a bio-process
2:     component.bioprocess ← new empty bio-process
3:     CompartsToBetaBinders (component)
4:     StatesToBetaBinders (component)
5:     SitesToBetaBinders (component)
6: end for
```

Algorithm 2. CompartsToBetaBinders (*component*)

```
1: binder ← new beta binder in component.bioprocess
2: binder.name ←"loc"                          ▷ the location is a beta binder
3: if component is in compartment at initial state then
4:     binder.type ← compartment.name^compartment.id          ▷ β(loc : cytosol_3)
5: end if
```

As Algorithm 3. shows, a beta binder is created to represent a state in its active/inactive form: its subject is the state name, while its type is the state name concatenated with the component name. In the example, LIFR can be a dimer (but it is a monomer at system initialisation), hence a binder $\beta(dimer : monomer\_LIFR)$ is added to $LIFR$ bio-process.

Algorithm 3. StatesToBetaBinders (*component*)

```
1: for all state ∈ component.states do          ▷ each state is a beta binder
2:     binder ← new beta binder in component.bioprocess
3:     binder.name ← state.name
4:     if component is in state at initial state then          ▷ β(dimer : dimer_LIFR)
5:         binder.type ← state.name^component.name
6:     else                                                    ▷ β(dimer : monomer_LIFR)
7:         binder.type ← state.opposite_name^component.name
8:     end if
9: end for
```

As Algorithm 4. shows, a beta binder is created to represent a site in its active/inactive form: its subject is the site name concatenated with the state name, while its type is the state name concatenated with the component name and the site name. In the example, site Y981 of receptor LIFR can be phosphorylated (but it is dephosphorylated at system initialisation), hence a binder $\beta(Y981\_pho : depho\_LIFR\_Y981)$ is added to $LIFR$ bio-process.

As Algorithms 5., 6. and 7. describe, each monomolecular event step is translated into one sequential pi-process which is placed into the bio-process representing the involved component, while each bimolecular event step is translated into two sequential pi-processes which are placed into the bio-processes representing the involved components.

Algorithm 4. SitesToBetaBinders (*component*)

for all $site \in component.sites$ **do** ▷ each site is a beta binder
  $binder \leftarrow$ new beta binder in $component.bioprocess$
  $binder.name \leftarrow site.name^{\wedge} site.state.name$
  if $site$ is in $state$ at initial state **then** ▷ $\beta(Y981\_pho : pho\_LIFR\_Y981)$
    $binder.type \leftarrow site.state.name^{\wedge} component.name^{\wedge} site.name$
  else ▷ $\beta(Y981\_pho : depho\_LIFR\_Y981)$
    $binder.type \leftarrow site.state.opposite\_name^{\wedge} component.name^{\wedge} site.name$
  end if
end for

Algorithm 5. EventToPiprocess (*event*)

if $event.reaction$ is relocation **then**
  $piproc \leftarrow$ chtype("loc", $new\_location\_type$)
else if $event.reaction$ is phosphorylation **then**
  $piproc \leftarrow$ chtype($binder\_site\_pho$, $site\_phosphorylated\_type$)
else if $event.reaction$ is dephosphorylation **then**
  $piproc \leftarrow$ chtype($binder\_site\_pho$, $site\_dephosphorylated\_type$)
else if ... **then**
end if
if $event.condition$ is specified **then**
  $piproc \leftarrow$ if $(binder\_cond, cond\_state\_type)$ then $piproc$
end if

Algorithm 6. EventToPiprocesses (*event*)

if $event.reaction$ is phosphorylation **then**
  $piproc1 \leftarrow \overline{binder\_event\_id}\langle\rangle$
  $piproc2 \leftarrow binder\_site\_pho()$.chtype($binder\_site\_pho$, $site\_phosphorylated\_type$)
  AddAffinity ($binder\_event\_id\_type, site\_dephosphorylated\_type$)
else if ... **then**
end if
if $event.condition$ is specified on $event.component1$ **then**
  $piproc1 \leftarrow$ if $(binder\_cond, cond\_state\_type)$ then $piproc1$
end if
if $event.condition$ is specified on $event.component2$ **then**
  $piproc2 \leftarrow$ if $(binder\_cond, cond\_state\_type)$ then $piproc2$
end if

The constructed pi-processes consist in sequences of communications and chtype operations, which can be prefixed by conditions on the type of binders to represent the specified conditions on events (Algorithms 5. and 6.). The names of bio-processes, pi-processes, beta binders and types follow a template, so that they are standardised, it is possible to refer to the previously defined beta binders, and no name clash occurs. Reaction rates and types affinities are assigned based on the input definitions. The constructed sequence of events represents a single

event step, so the reaction rate is assigned to the first action, while the others are assigned infinite rates (so that after the first one occurs, the others are immediately executed). The actual sequence of actions depends on the reaction type. In the example, event 17 involves the phosphorylation of sites Y917 and Y945 on OSMR, translated into a sequence of two chtype actions. Moreover, the phosphorylations can occur only if OSMR is a dimer, hence the sequence is prefixed by a condition on the beta binder representing the dimer state. The final pi-process, added to $OSMR$ bio-process is, therefore,

$$Pho\_17 = \text{if } (dimer, dimer\_OSMR) \text{ then } \mathsf{chtype}(0.2,\ Y917\_pho,\ pho\_OSMR\_Y917)\ .\ \mathsf{chtype}(inf,\ Y945\_pho,\ pho\_OSMR\_Y945)\ .\ Pho\_17\ .$$

The ordering of the events is given in the following way (Algorithm 7.). If two reactions are concurrent, they are translated into processes placed in parallel composition. If, instead, they have to be executed one after the other, the second one is prefixed by a condition on the type of the binder which is modified at the end of the first one (satisfied if the type is the one modified by the first reaction). If, finally, they are mutually exclusive, both events are prefixed by a condition on the type of the binder which is modified at the end of the other one (satisfied if the type is not the one modified by the other reaction).

Algorithm 7. NarrativeToPiprocesses

```
for all ev ∈ Events do                                   ▷ each event is one or two sequential pi-processes
    if ev is monomolecular then                          ▷ one pi-process is inserted into the bio-process
                                                           of the involved component
        pproc ← EventToPiProcess (ev)
        if ev is alternative to prev_ev then
            pproc ← if not(prev_ev.mod_binder, prev_ev.mod_type) then pproc
            prev_ev.pproc ← if not(ev.mod_binder, ev.mod_type) then prev_ev.pproc
        end if
        AddPiprocessInParallel (ev.component.bioproc, pproc)
    else if ev is bimolecular then                       ▷ one pi-processes is inserted into the bio-process
                                                           of each involved component
        ⟨ pproc1, pproc2 ⟩ ← EventToPiProcesses (ev)
        if ev is alternative to prev_ev then
            if both ev and prev_ev involve ev.component1 then
                pproc1 ← if not(prev_ev.mod_binder, prev_ev.mod_type) then pproc1
                prev_ev.pproc ← if not(ev.mod_binder, ev.mod_type) then prev_ev.pproc
            end if
            if both ev and prev_ev involve ev.component2 then
                pproc2 ← if not(prev_ev.mod_binder, prev_ev.mod_type) then pproc2
                prev_ev.pproc ← if not(ev.mod_binder, ev.mod_type) then prev_ev.pproc
            end if
        end if
        AddPiprocessInParallel (ev.component1.bioproc, pproc1)
        AddPiprocessInParallel (ev.component2.bioproc, pproc2)
    end if
end for
```

3.3 Case Study: The Translation of the Gp130 Signalling Pathway Model

A prototype of the tool has been developed and integrated into BetaWB. The model described in Sect. 2.1 was translated into Beta-binders by using this

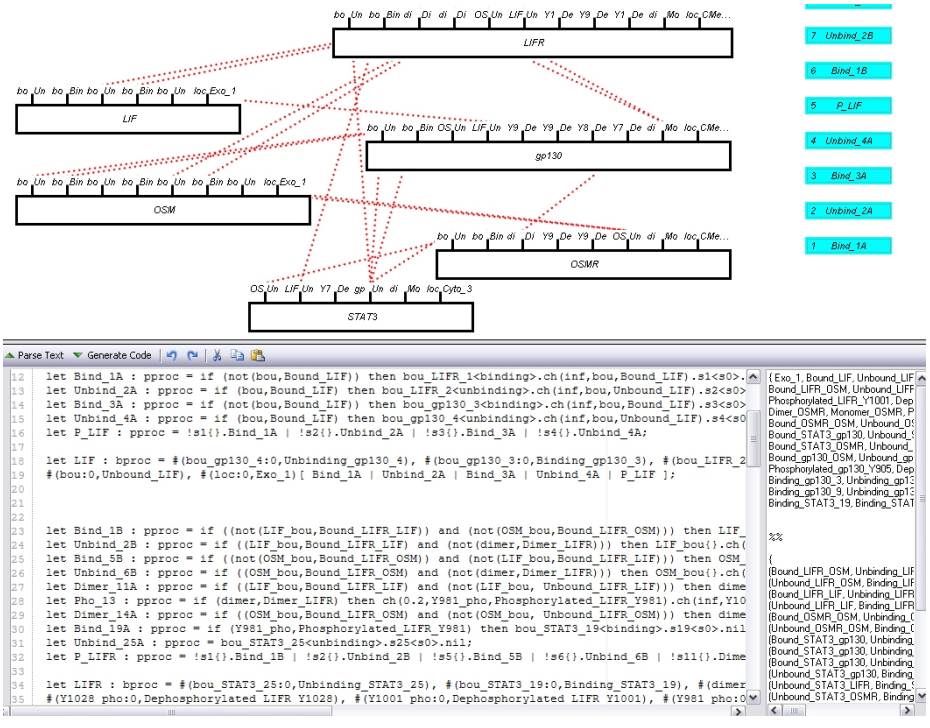


Fig. 1. The Gp130 pathway Beta-binders model imported in BetaDesigner

prototype, and then simulated by using BetaWB. Figure 1 is a BetaDesigner screenshot showing the graphical visualisation and part of the imported Beta-binders code which has been obtained from the translation of the Gp130 pathway model.

We do not present in this work any biologically relevant result: in order to achieve this final goal, more details on the pathway dynamics should be taken into account and more precise information on reaction rates should be acquired. Moreover, some aspects of the translation should be improved for the tool to be practically used for translation of complex models. This work was primarily meant to be a sort of feasibility study. We believe that the proposed language, together with the automatic translation into Beta-binders and the existing simulator, allows the modeller to describe biological systems in simple words, simulate the model, and obtain sensible results. The Gp130 model we have described in this work was developed by a biologist who had no previous experience in modelling. This gives us some confidence that our main goal, which is to have a user-friendly language which biologists feel comfortable with, was achieved.

4 Conclusions and Further Work

The proposed modelling language and the automatic translation into a formal language allow us to hide the formal details from life scientists. Therefore, we believe that life scientists could easily use the textual language to describe systems, and automatically obtain simulation and analysis results.

The choice of which primitives had to be included in our language has been done to have a simple and basic set of events which could be described. The language and the tool can be extended with new constructs and some improvements on the already present ones can be done. The Beta-binders code obtained from the translation is quite intuitive; however, the usage of other Beta-binders features which could make the translation even more intuitive and efficient (e.g. events [10] and biological transactions [15]) is under investigation.

Several description languages (both textual and graphical) have recently been proposed to model biological systems. A formal comparison with those description languages will be done, and the interchangeability with graphical notations will also be taken into account.

Finally, we believe that in order to fully benefit from this approach an integration with some of the many existing biological databases would be much useful. This would allow, for example, the automatic extraction of data and parameters from those databases. In addition to this, another interesting aspect is the automatic extraction of information from the simulation output to be used again to refine the input model.

Acknowledgments

The authors wish to thank Nicholas Underhill-Day (Cancer Research UK) for his help with the Gp130 model.

References

1. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
2. Priami, C., Regev, A., Silverman, W., Shapiro, E.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters 80(1), 25–31 (2001)
3. Phillips, A., Cardelli, L.: A Correct Abstract Machine for the Stochastic Pi-calculus. In: BioConcur '04, Workshop on Concurrent Models in Molecular Biology (2004)
4. Priami, C., Quaglia, P.: Operational patterns in Beta-binders. Transactions on Computational Systems Biology 1, 50–65 (2005)
5. Calder, M., Gilmore, S., Hillston, J.: Modelling the Influence of RKIP on the ERK Signalling Pathway Using the Stochastic Process Algebra PEPA. T. Comp. Sys. Biology 7, 1–23 (2006)
6. Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O., Heath, J., Gaffney, E.: Simulation and verification for computational modelling of signalling pathways. In: Proc. Winter Simulation Conference, Omnipress, pp. 1666–1675 (2006)

7. Kholodenko, B.: Cell signalling dynamics in time and space. Nature Reviews Molecular Cell Biology 7(3), 165–176 (2006)
8. Kohn, K.: Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems. Molecular Biology of the Cell 10, 2703–2734 (1999)
9. Kitano, H.: A graphical notation for biochemical networks. Biosilico 1(5), 169–176 (2003)
10. Romanel, A., Dematté, L., Priami, C.: The Beta Workbench. Technical Report TR-03-2007. Technical report, The Microsoft Research - University of Trento Centre for Computational and Systems Biology (2007)
11. Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., Kitano, H.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19, 524–531 (2003)
12. Underhill-Day, N., Heath, J.: Oncostatin M (OSM) Cytostasis of Breast Tumor Cells: Characterization of an OSM Receptor β-Specific Kernel. Cancer Research 66(22), 10891–10901 (2006)
13. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Molecular biology of the cell. Garland Science (2002)
14. Cardelli, L.: On Process Rate Semantics (2007), Available at http://lucacardelli.name/Papers/On%20Process%20Rate%20Semantics.pdf
15. Ciocchetta, F., Priami, C.: Beta-binders with Biological Transactions. Technical Report TR-10-2006. Technical report, The Microsoft Research - University of Trento Centre for Computational and Systems Biology (2006)

Expressive Models for Synaptic Plasticity*

Andrea Bracciali[1], Marcello Brunelli[2], Enrico Cataldo[2], and Pierpaolo Degano[1]

[1] Dipartimento di Informatica, Università di Pisa, Italia
[2] Dipartimento di Biologia, Università di Pisa, Italia
braccia@di.unipi.it, mbrunelli@biologia.unipi.it, ecataldo@biologia.it, degano@di.unipi.it

Abstract. We explore some presynaptic mechanisms of the *calyx of Held* synapse through a stochastic model. The model, drawn from a kinetic approach developed in literature, exploits process calculi as formal grounds, enjoys nice compositional properties, has a direct computational implementation that supports simulation trials, and, to our knowledge, represents the first process calculi based model of a presynaptic terminal. Simulation results have shown coherence with experimental data and robustness against sensitivity analysis. The core model has been extended in order to address some issues related to open problems: we discuss hypotheses on short-term synaptic enhancement (facilitation) and depression, i.e. plasticity mechanism that are related to memory and learning. The two aims of our work, i.e. addressing neural mechanisms and validating and possibly improving, process calculi based modeling techniques are discussed throughout the paper, together with the results of experiments.

1 Motivations

Research in life sciences is benefiting from a large availability of formal description techniques and analysis methodologies. These allow both the phenomena investigated to be precisely modeled and virtual experiments to be performed *in silico*. Such experiments may result in easier, faster, and satisfying approximations of their *in vitro/vivo* counterparts. A promising approach is represented by the study of biological phenomena as a collection of interactive entities through process calculi equipped with stochastic semantics. These exploit formal grounds developed in the theory of concurrency in computer science, account for the not continuous, nor discrete, nature of many phenomena, enjoy nice compositional properties and allow for simulations that have been demonstrated to be coherent with data in literature. The huge amount of information produced in the field of neurobiology and the complex dynamics of the biological processes require the utilization of mathematical and computational modeling methods [15,8]. Neurons represent the elementary components of the nervous systems, able to communicate with each other at highly specialized contact sites called *synapses*.

* This work has been partially supported by the MIUR project *Bisca*.

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 152–167, 2007.

In general, each neuron consists of a somatic cellular body, on which a variable number of thin elongated structures called *dendrites* converge and from which a long single structure called *axon* emerges, branching in several synaptic terminals. The synaptic terminals of the transmitting neuron (the presynaptic element) send signals by releasing chemical molecules (*neurotransmitters*) to the dendritic part of the receiving neuron (*postsynaptic term*) [1].

The synapses are the places of functional contacts between neurons, where the information is stored and transmitted from one to another neuron. Synaptic transmission is a complex process and current knowledge on synapses is based on the analysis of a limited number of experimental synaptic models [10]. Synaptic transmission involves the presence of calcium ions in the presynaptic terminal, which control the transmitter release process [38], consisting in the exocitosis of synaptic *vescicles* (small elements containing the neurotransmitters) located at the presynaptic so-called *active zone* [35].

The electrical signals (action potentials) arriving at the synaptic terminal induce the opening of the Ca^{2+} channels. The transient elevation of the internal Ca^{2+} concentration in the presynaptic terminal triggers synaptic vescicle exocitosis, and hence the neurotransmitter release (calcium-triggered-release hypothesis). Interestingly, chemical messengers (intracellular) and modulators (extracellular) regulate the relationship between action potential and release in a synaptic terminal, which is also altered by the repeated activity. All these things make the presynaptic terminal a kind of computational unit, which changes its output based on its previous activity and ongoing modulation.

Theoretical and functional studies have suggested that Ca^{2+} acts on presynaptic vescicles by a local huge and short-lived elevation of its concentration. The locality and rapidity of the concentration variation render the study of this phenomenon not approachable with the conventional microscopic imaging techniques. Among the methods envisaged to overcome this limitations, one very fruitful is the so-called reverse approach, in which Ca^{2+} uncaging is induced in the presynaptic element. The uncaging method induces spatially homogeneous Ca^{2+} elevation, implying that measuring the Ca^{2+} fluorescent indicator gives an indication of the real Ca^{2+} sensed by the vescicles. This experimental method has been applied to the study of the large synapse of the auditory tract of the central nervous system, called *calyx of Held*. Moreover, it has been possible to build a minimal kinetic model for the process of the Ca^{2+} triggered vescicle release. Also, one can infer local Ca^{2+} signal waveform which is compatible with the experimental data on the time course and amplitude of release [30,31].

Most of the models treating the calcium triggered release issues present some methodological limitations. These deterministic models, and among them the *calyx of Held* model, use differential equations, known as reaction rates equations, to describe the time course of $[Ca^{2+}]$, the Ca^{2+} concentration ($mole \times liter^{-1}$) interacting with the synaptic vescicles. This approach implies that $[Ca^{2+}]$ is continuous, while it is clearly not [17]. For example, with a Ca^{2+} concentration of 10 μM in a volume of 60 nm^3 there is a single free ion. Another common assumption is that the binding of the Ca^{2+} to the release sensor of the vescicle

does not affect the $[Ca^{2+}]$ concentration [17]. Also this assumption is not properly adequate: considering that the dimensions of the vescicle diameters range in the interval 17-22 nm, in a volume of 60 nm^3 there could be few Ca^{2+} ions, and when some of them bind to the vescicle sensors, the number of calcium ions could change substantially.

In general, when the fluctuations in the molecular population levels are relevant, e.g. when the numbers per unit volume of the molecular species involved are small, the stochastic approach has to be preferred. The fluctuations, that can be seen as random variations about a mean number of molecules, can significantly alter the dynamics of biochemical pathways. Just to cite few examples, fluctuations might decrease or increase the steepness of a nonlinear stimulus-response relationship. In other cases, via the so-called stochastic resonance, they can increase the reliability of response to small signals [33].

Hence, in these cases, the use of a stochastic approach appears to be much more appropriate, since the deterministic approach fails to capture the nature of chemical kinetics, which at low concentrations is discrete and stochastic [37].

In the stochastic approach, the system is described by the so-called "master equation", which usually is intractable. A stochastic simulation algorithm has been proposed in [11] to overcome these difficulties.

Recently, stochastic techniques have been also adopted in computer science to model quantitative aspects of interactive systems within concurrency theory. Concurrency theory aims to model the behaviour and the structure of systems composed of autonomous computational entities, which dynamically interact one with another, possibly reconfiguring the system itself. At the beginning, stochastic models have been used to study performance/time related properties, e.g. [12]. The strong analogies between concurrent and living systems, "cells as computation" [25], has fostered the development of Systems Biology [14,6], a systemic approach to living system modeling.

According to this metaphor, cells, molecules and biological "active" components, i.e. those capable of exhibiting a behaviour, are assimilated to computer processes, the computational units of a concurrent software system. Then, biological interaction corresponds to process communication. By communicating, processes may exchange information or synchronise themselves, i.e. they interact one with another. Finally, a biological experiment, or biological activity in general, has then a direct correspondence into computation. That is, biological processes can not only be simulated by *in silico* experiments, but it is also possible to formally reason about their computational models and infer properties of interest. *Process calculi* are a formalism to describe such models: systems are *compositionally* described in terms of suitable *abstractions* of their component behaviour. Several process calculi whose "operators" are oriented to describing different aspects of biological interaction have been proposed e.g. [19,23,24,5]. Some of these calculi have been equipped with stochastic semantics in order to study the quantitative evolutions of systems, e.g. [22,23,16,4]. This approach benefits from conjugating the abstract and compositional algebraic models, the possibility of precisely describe their semantics and formally reasoning about

them, and the quantitative analysis provided by stochastic semantics. Executable implementations of the calculi and analysis tools are provided.

In this context, motivated by addressing some aspects of the functioning of neural synapses, we have developed a stochastic model of the calcium triggered release in the *calyx of Held* synapse.

Our work starts from a deterministic model presented in [30], from which we have derived a suitable stochastic model. This has subsequently been formalised in a variant of the Pi-calculus [18], in which the behaviour of Ca^{2+} and vescicles has been described and composed to form the presynaptic terminal. Model development has benefited from the above mentioned features, like modular design, abstract representation of the component functioning and stochastic interpretation of the system dynamics. Then, *in silico* experiments have been carried out by means of the stochastic Pi-calculus simulator SPiM [21], which represents one of the most complete and expressive simulation environment for stochastic calculi currently available. Beyond on the availability of a suitable execution environment, the choice of Pi-calculus has been based on its expressiveness. Although not all its distinguishing features have been exploited here, like the dynamic creation of new interaction capabilities (new names), it has seemed reasonable not to renounce to them, which might reveal useful when further detailing the model. The definition of the "right" representation language still appears as an open problem.

The developed model has been firstly tested against sensitivity and robustness, then tuned for our experiments of interest and, finally, used to investigate two aspects of the presynaptic plasticity mechanisms: paired pulse facilitation and short term depression.

Obtained results are coherent with those in literature and appear useful for better understanding and testing hypothesis on not fully understood parts of the two addressed issues.

Taking advantage of the compositionality of our approach, assemblying the more detailed neural model has been quite straightforward and, also, has suggested interesting directions for the enhancement of the expressive power of the representation language.

In the long term, we are interested in further pursuing the investigation along two complementary paths: from the biological viewpoint, we are interested to address models of synaptic plasticity, i.e. activity dependent change mechanisms, which are the bases of memory and learning processes, and to build more comprehensive stochastic models of synaptic functioning; from the computer science viewpoint, we aim at further developing the theory of concurrent biological interaction. In this sense, addressing a notion of (spatial) locality along the line suggested by the experiments would be a challenging task.

Synopsis. The proposed model is described in Section 2, experiment results and a discussion on the model are presented in Section 3. Related works are discussed throughout the paper, while Section 4 contains concluding remarks. Preliminary results of our work have appeared in [3].

2 A Process Calculi Based Stochastic Model

In this paper we have applied a stochastic model, based on the algorithm introduced in [11], to describe the calcium triggered release mechanisms studied in the model system of the synapse *calyx of Held*. Our staring point was a phenomenological kinetic model, described in [30], in which five calcium binding steps and a cooperativity factor b are needed for vescicle activation and release. The kinetic model parameters were computed by fitting experimental data, obtained by means of elevating the intracellular presynaptic $[Ca^{2+}]$ in a controlled, homegeneous and step-like manner (calcium uncaging) [30]. We transformed the equations of the above cited model by utilising the relationship between the stochastic rate constants (c) and deterministic rate constants (k) [13]. For reactions of the first order, $c = k$. For reactions of second order, the relationship becomes: $c = k/(NA \times Vol)$, where NA represents the Avogadro's number and Vol the volume of the reaction. Hence, in order to determine the values of the stochastic rate constants, we needed to estimate the value of Vol. Spatially, the *calyx of Held* is organized as a "parallel" arrangement of a large array of active zones, ranging from 300 to almost 700 [29]. Active zones, each containing up to 10 vescicles, are clustered in groups of about 10 of them, in a volume having a diameter of almost 1 μm. Each action potential activates all the active zones. Such particular morpho-functional organization of this synapse has allowed us to model a subunit of the presynaptic element, consisting of a cluster of 10 active zone, each containing 10 vescicles, in a volume of 0.5 10^{-15} $liter$. With this volume estimate, we have obtained the following values for the stochastic constants: $c_{on} = 9 \times 10^7 \ / \ (6.02 \times 10^{23} \times 0.5 \times 10^{-15}) \ s^{-1} = 0.3 \ s^{-1}$, $c_{off} = 9500 \ s^{-1}$, $\gamma = 6000 \ s^{-1}$ and $b = 0.25$, and the following numbers of Ca^{2+} ions: 300, 3000 and 6000, corresponding to molar concentrations $[Ca^{2+}]$ of 1, 10 and 20 μM. The equations of the stochastic model are:

$$Ca_i^{2+} + V \underset{c_{off}b^0}{\overset{5c_{on}}{\rightleftarrows}} V_{Ca_i^{2+}} + Ca_i^{2+} \underset{2c_{off}b^1}{\overset{4c_{on}}{\rightleftarrows}} V_{2Ca_i^{2+}} + Ca_i^{2+} \underset{3c_{off}b^2}{\overset{3c_{on}}{\rightleftarrows}} V_{3Ca_i^{2+}} + Ca_i^{2+}$$

$$V_{3Ca_i^{2+}} + Ca_i^{2+} \underset{4c_{off}b^3}{\overset{2c_{on}}{\rightleftarrows}} V_{4Ca_i^{2+}} + Ca_i^{2+} \underset{5c_{off}b^4}{\overset{c_{on}}{\rightleftarrows}} V_{5Ca_i^{2+}} \xrightarrow{\gamma} T$$

where Ca_i^{2+} represents the number of intracellular calcium ions, V the number of vescicles, T the released vescicle.

Our simulations have confirmed the results obtained with the deterministic model: high sensitivity of vescicles to calcium concentrations [30]. While several other synapses require a calcium concentration in the range of 100-300 μM for triggering vescicle release [35], it is known that the local calcium concentration can be much lower than 100 μM in the *calyx of Held* [30]. Our results have also confirmed this result, by showing that concentrations as low as 1, 10 and 20 μM are able to deplete the releasable pool in a few milliseconds. In the following, we will graphically report our results. Each figure consists of three pictures: on

the left side the time course of Ca^{2+} is displayed (together with the extrusion mechanism (`P`) and its occupancy (`CaP`) when present, and the intermediate Ca^{2+} bindings); in the middle the same picture is shown in logarithmic scale so as to appreciate the intermediate states of calcium binding and vescicle activation (`Vstar`) and release (`T`); on the right side the focus is on activated vescicle (`Vstar`) and the total number of the transmitter released (`T`).

Figure 1 shows simulation results for the step-like calcium case with the following parameters: $V = 100$; $Ca^{2+} = 6000$; $c_{on} = 0.3$; $c_{off} = 9500$; $\gamma = 6000$; $b = 0.25$. It can be observed that the pool of vescicles is 80% depleted within 3 ms, coherently to the experimental findings [30].

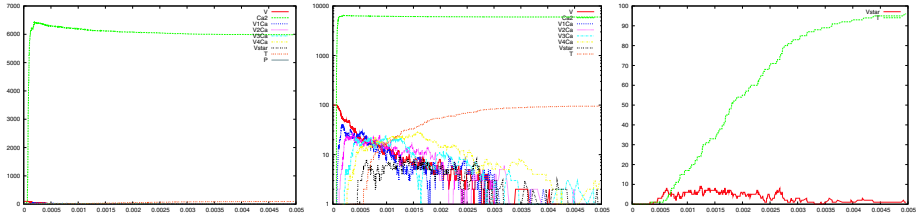


Fig. 1. Step-like calcium uncaging (V=100; Ca=6000; C_{on}=0.3; C_{off}=9500; γ=6000; b=0.25)

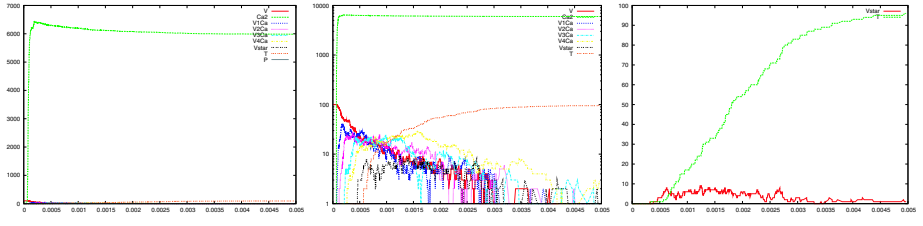


Fig. 2. Wave-like calcium uncaging (V=100; Ca=6000; C_{on}=0.3; C_{off}=9500; γ=6000; b=0.25)

The experiments and models on Ca^{2+} uncaging [30] showed a high sensitivity of vescicle release in response to a uniform elevation of $[Ca^{2+}]$ in the range 10 μM. It was not clear whether very short $[Ca^{2+}]$ elevations are sufficient to induce a release similar to that induced during an action potential. A recent experimental work [2] has addressed this issue. A spatially uniform and very rapidly decaying $[Ca^{2+}]$ transient, obtained by Ca^{2+} uncaging in the presence of added Ca^{2+} buffers, was induced in the presynaptic element of a *calyx of Held* synapse. This short-lived elevation (wave-like) of calcium concentration has been revealed to be able to trigger vescicle release. We have introduced in our model a simple mechanism of calcium extrusion utilized in a previously developed model [7], adapting the rate constants to fulfill our needs:

$$Ca_i^{2+} + P \underset{c_2}{\overset{c_1}{\rightleftarrows}} CaP \xrightarrow{c_3} Ca_o^{2+}$$

where Ca_o^{2+} is the extruded calcium, P is an abstraction of a pumping mechanism, $c_1 = 8\ s^{-1}$, $c_2 = 25\ s^{-1}$ and $c_3 = 10000\ s^{-1}$. We have obtained a simulated calcium wave lasting about 1 ms and with a half width 0.5 ms, conforming to the experimental requirements [2]. Such a kind of calcium wave with a peak value of about 6000 ions, corresponding to a peak calcium concentration of 20 μM, can be seen on the left-side of Figure 2. In the right side of the same figure, the release of one vescicle can be observed. Considering that a whole presynaptic element can be made of about 70 of our simulated clusters, this implies that a single action potential, and accordingly a single calcium wave, is able to release a significant amount of vescicles. This is also along the line of the experimental findings [30,31,2].

For both models (step-like and wave-like calcium), we have performed a parameter variation study (sensitivity analysis). We have run simulations for different values of the number of vescicles (reference value 100, other values: 10, 50, 200 and 500), the number of calcium ions Ca^{2+} (reference values 300, 3000 and 6000, others: 12000, 18000, 24000), the stochastic coefficients c_{on} (reference value 0.3, others: 0.1, 0.2, 0.4 and 0.5), c_{off} (reference value 9500, others: 5500, 7500, 11500 and 13500), b (reference value 0.25, others: 0.1, 0.2, 0.3 and 0.4) and γ (reference value 6000, others: 2000, 4000, 8000 and 10000).

One of the results of this analysis is that the forward coefficient c_{on} seems to have a critical role: higher values of this coefficient correspond to a faster release, in the step-like case, and to a switch from no-release to a consistent release, in the wave-like case. For the step-like case, we showed that a simple variation of the coefficients c_{on} and b changes the dynamics of the release processes in an unpredictable manner.

This kind of experiments are of interest when addressing the problem of the variations in the release rate, one of the still obscure phenomena which have been observed about vescicle release. These variations have been explained by the recruitment of new vescicles within the same active zone or by a different sensitivity to calcium ions of the vescicle belonging to same cluster. Our kind of analysis might give some contributions to the debate on the interpretation of these controversial experimental data [30,31,2].

2.1 Neuro-processes

In order to illustrate the main features of the formal model used to stochastically represent the behaviour of the *calyx of Held* synapse, we briefly sketch some of its parts. Excerpts from the model, viz. its implementation for the SPiM stochastic interpreter [21], are in Figure 3. The representation language models interaction as pairs of input/output actions over the same communication channel (`?c`/`!c`). These atomic actions can be composed in a sequence (`;`) or in alternative choices (`?c or ?d`) so as to form a process (`p()= ...`). Processes can run in parallel

(`p()|q()`). Initially, the length of the simulation is set (here $0.005s$), then some stochastic parameters are defined. Communication channels can be (dynamically) created by means of the `new` command and have associated a stochastic rate (this and the current quantities of reactants determine the probability of a reaction "happening" through the channel). A calcium ion (`ca()`) can interact with a vescicle (`v()`) over channel `vca` with rate `con5=1.5` (beyond being able to do other things). After this communication, `ca()` disappears and `v()` becomes `v_ca()`, representing the binding of the two. This realises a second order reaction.

```
directive sample 0.005

val con5 = 1.5
val b = 0.25
val coff5 = 47500.0 * b * b * b * b

new vca@con5:chan

ca() =  do ?vca;()
        or ?v2ca;()
        ...
        or ?cp;()

v() =   !vca;  v_ca()

v_ca() = do !bvca; v()
         or !v2ca; v_2ca()

Dv_ca() = ?bvca; ( ca() | Dv_ca() )

w(cnt:int) = do delay@40000.0;
                if 0 <= cnt then ( 80 of ca() |  80 of w(cnt - 1)) else ()
             or !void; ()

run   1 of w(1)   1000 of p()   100 of v()   1 of (Dv_ca() | Dv_2ca()| ... )
```

Fig. 3. The *calyx of Held* SPiM code

First order reactions are modeled as interactions with a *single* dummy molecule (so as not to alter stochastic dynamics). For instance, `v_ca()` can then either accept other calcium bindings or degradate back to an unbound vescicle by communicating through `bvca` with `Dv_ca()`, which restores `ca()` and itself. So far, the system has been described by specifying simple atomic behaviour, basically corresponding to chemical reactions, and then composing them together. The parametric process `w(cnt:int)` allows us to suitably modulate the calcium wave. After a stochastic delay, it replicates 80 copies of itself in parallel with 80 `ca()` if its integer parameter `cnt` is positive, otherwise it dies. This realises an exponential growth, which can be controlled by the delay rate and the parameter in its rapidity and quantity. Finally, the initial state can be populated specifying

how many molecules of each specie are present (1 wave, 1000 pumping molecules, 100 vescicles and 1 copy of the needed dummy molecules).

3 Results and Model Evaluation

Exploiting the developed model, we have addressed some open issues regarding the presynaptic mechanisms for Ca^{2+} triggered vescicle release. More specifically, we have considered *temporal* and *spatial* aspects of the release which appear intertwined and relevant for synaptic *plasticity*. On the one hand, we have studied the behaviour of the synaptic terminal in the presence of a train of action potentials occurring within short temporal intervals. We report on an analysis of two Ca^{2+} waves and discuss how this is related to *facilitation*, i.e. a form of activity dependent enhancement of the synaptic strength. On the other hand, we have tried to find a suitable model of the spatial (and functional) distribution of vescicles within the presynaptic terminal. This has been done in order to support the verification of competing hypotheses on the mechanisms ruling *short-term synaptic depression*. With this aim we have analysed the relationship between the measured pattern of the time course of release and the possible existence of two pools of different vescicles in the active zone [9,28,36,27].

In order to tune the model for these two experiments, we have beforehand run a series of simulations (not reported here) with the aim of both evaluating the variance of the results of different runs, and of tuning the model to the hypotheses of the experiments. In particular, the hypotheses about vescicle distribution has required us to revise the assumption on the number, and also the behaviour, of vescicles. This is needed to obtain average release values during a single action potential that are coherent with the experimental findings in the scenaria of interest. In particular, setting the number of the readily releasable vescicles (V) to 50 and increasing their propensity to calcium binding (c_{on}) to $0.4s^{-1}$ has resulted in an average number of released vescicles of about 2, which is about 4% of the (readily releasable) vescicle pool, adherent to the experimental findings [9].

The compositionality of the chosen representation language has permitted us to easily embed the new features in the model by means of modular changes. Moreover, the challenge of dealing with the illustrated spatial and temporal aspects seems to suggest fruitfully directions for extending the expressiveness of the representation language. Both these aspects are discussed in Section 3.3.

With this more detailed model, we have hence studied the phenomenon of synaptic transmission called paired pulse facilitation (Section 3.1) and synaptic depression during sustained depolarization of the synapse (Section 3.2), which are examples of short-term synaptic plasticity still puzzling neuroscientists. In the following, we report the simulations that better illustrate our results, out of a more detailed statistical analysis which is scope for future work.

3.1 Paired Pulse Facilitation

The term synaptic facilitation indicates a form of activity-dependent synaptic enhancement observed in several synapses, in which the synaptic strength

increase during a train of action potentials [9,38]. In general, facilitation depends by pre- and post-synaptic mechanisms. In this paper, we focus on the presynaptic side of facilitation. It is generally accepted that the intracellular Ca^{2+} remaining from previous activity (the so-called *residual* Ca^{2+}) causes facilitation, but the mechanisms that make this happen are still not clearly understood [35]. In general, the residual Ca^{2+} is less than 1 μM (in our model this correspond to less than 300 Ca^{2+} ions) and for most synapses the local $[Ca^{2+}]$ increase needed for release is between 100 and 300 μM. This implies that, typically, the residual Ca^{2+} is not sufficient to induce facilitation.

Some other underlying mechanisms for facilitation have been proposed, such as, just to cite some, the permanence of Ca^{2+} bound to the high affinity binding

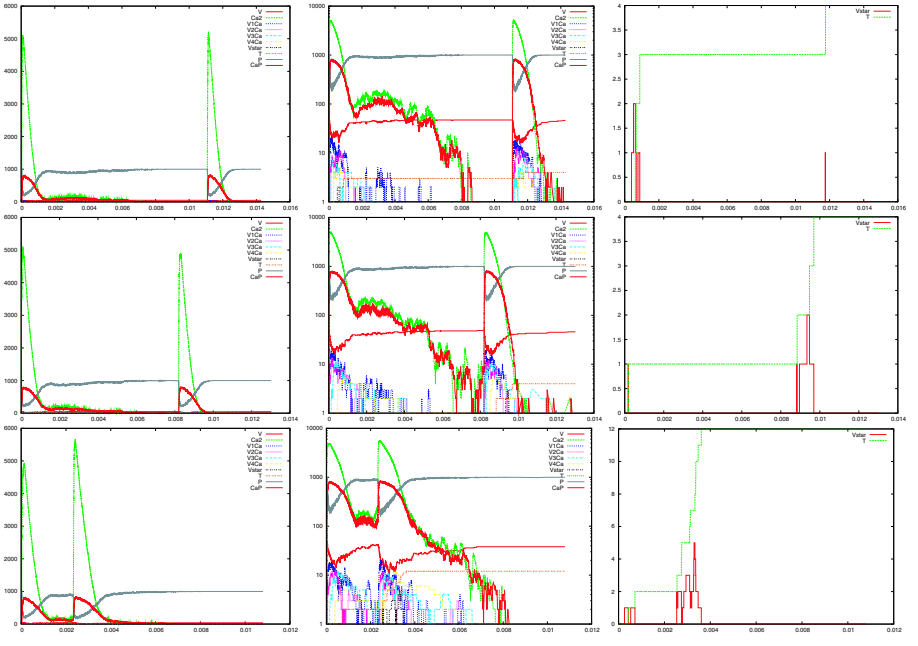


Fig. 4. Two wave-like calcium pulses at different intervals (V=50; C_{on}=0.4)

sites of the secretory machinery, Ca^{2+} buffer saturation or an increased size of the readily releasable (see Section 3.2) pool [9].

We have run a series of simulations in which we have used a double Ca^{2+} wave in order to study facilitation. We report in Figure 4 a small sample selection of our simulations with varying time delays between waves, each row represents a simulation. In the left pictures, and correspondingly in the others, it is possible to appreciate the occurrence of the two waves with varying delays. From upper to lower row, the time interval betwee Ca^{2+}wave decreases (12, 8 and 2 ms) and it can be observed that for smaller delay the amount of release due to the second wave increases notably.

The possible underlying mechanisms could be suggested by observing the central column of Figure 4. Just before the second calcium wave develops, the amount of residual $Ca^{2+}$is bigger when the delay decreases. This causes a concomitant increase of the release.

At the same time, it seems that other parts of the release machinery could be involved in facilitation, such as the occupancy of P or the intermediate steps of vescicle binding. In our simulations, also these parts seem to be influenced by residual Ca^{2+}, as can be seen, for instance, by looking at the level of occupancy of P. Hence, the residual Ca^{2+} seems to have a central role in the facilitation process, even at very low concentrations.

Summing up, our simulations support the hypothesis that in *calyx of Held*, facilitation is likely due to residual Ca^{2+} and to occupancy of Ca^{2+}buffers, which are cellular elements which control, by reducing it, the Ca^{2+} concentration. Our simulations show that our model is consistent with the idea that a very low level of residual Ca^{2+} might account for the particular form of short-term synaptic plasticity named paired pulse facilitation [9,38].

3.2 Short-Term Synaptic Depression

It is known by experimental data that the vescicles of the active zone can be divided in two virtually separated and equally populated pools: one consisting of so-called readily releasable vescicles and the other one of so-called reluctantly releasable vescicles [20,28,36].

During a single action potential discharge, which we represent as a single calcium wave, the vescicles released belong to the readily releasable pool. The amount of release corresponds to about 4% of the readily releasable pool, hence in our model to 2-3 vescicles [9]. Whereas, when the synapse is depolarized to elicit maximal Ca^{2+} influx, all the vescicles of the active zone are released (readly and reluctantly releasable) and the synapse active zone is completely depleted. In this case the time course of the release process is charachterized by two time constants, which are about 3 and 30 ms, during which the readily and reluctant vescicle are released, respectively [20,32]. It is also known that the reluctant vescicles are replaced much more rapidily than the readily releasable ones.

The precise mechanisms through which this form of release happens are still debated. For some cells, such as the chromaffin cells, it is likely that the reluctant vescicles differ, in their intrinsic kinetics, from the readily releasable ones [34].

For the *calyx of Held* synapse, it has been suggested, by means of a qualitative model, that the reluctant vescicles are precursors of the rapidly releasing ones and that they become readily releasable by laterally moving toward the readily releasable pool [20]. Based on these suggestions, we have built a stochastic multi-pool model of the possibile mechanisms underlying short-term synaptic depression. The coefficients have been in part obtained from the literature and in part obtained by trails and errors, because generally unknown.

In this stochastic model, one pool represents the readily releasable vescicles (docked to the active zone), which are released according to the model previously developed and whose starting number has been set to 50.

The reluctant vescicles (whose starting number is 50) go back and forth between two other pools, in which they are undocked (Rct_V), i.e. not releasable, and docked (R_V), i.e. releasable, respectively. The undocked reluctant are rapidily replenished from a forth pool (Inf_V) of vescicles, which represents the reservoir of the reluctant vescicles:

$$Inf\_V \underset{1s^{-1}}{\overset{10s^{-1}}{\rightleftarrows}} Rct\_V \underset{0.1s^{-1}}{\overset{2.5s^{-1}}{\rightleftarrows}} R\_V + Ca_i^{2+} \underset{c_{off}b^0}{\overset{5c_{on}}{\rightleftarrows}} \cdots \underset{c_{off}b^4}{\overset{c_{on}}{\rightleftarrows}} R\_V_{5Ca_i^{2+}} \xrightarrow{\gamma} T$$

The results of the simulations are reported in Figure 5 for the case of 300 and 1000 Inf_V vescicles, respectively. Here, the left picture shows calcium dynamics as usual, the center one reports calcium again and the four kind of vescicles and their activation and release in logarithmic scale, while the rightmost picture reports again activated vescicles and the corresponding cumulative release. In both simulations, the continuous depolarization was mimicked by a step like calcium wave. In the central column it can be seen the synchronous release (darker, within the first 3 ms) and the asynchronous one (lighter in the picture). In the right pictures, it is very clearly visible the double slope of the cumulative release curves. The two right pictures are qualitatively similar: it can be observed a fast release (during about 3 ms) of about 50 vescicles, followed by a slower release. In the first row of Figure 5, during the slower release, 50 vescicles (rather than 150 of the second row) are released in about 30 ms, which is most adherent to the experimental findings.

This model helps to clarify the mechanisms of delocalization of vescicles in the active zone, helps to determine some unknown parameters (stochastic coefficients, number of vescicles) and reproduces some experimental observations. It must be noted that it is a partial description of the complex and still poorly understood short-term synaptic depression process [35]. Nevertheless, it suggests

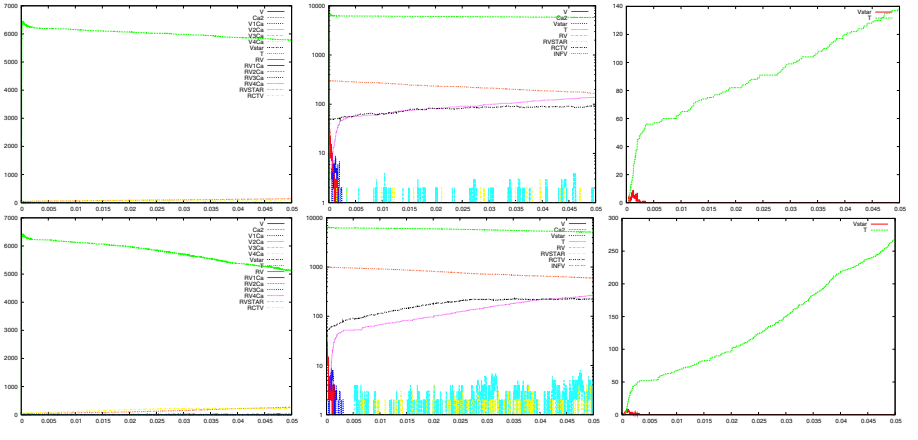


Fig. 5. Multipools vescicle activation (C_{on}=0.4; V=50; R_V=0; Rct_V=50; Inf_V=300(top); Inf_V=1000(bottom))

that the stochastic constants cannot be effectively constant: they would need to be modulated during the ongoing process of release, otherwise the movements of the vescicle would be out of equilibrium condition during the rest state of the synapse (see for example [34,26]). This last condition could explain the discrepancy between the values of some coefficients utilized in the model and the corresponding values found in the literature, for example the value 2.5 for the coefficient was indicated to be 0.25.

3.3 Extending the Stochastic Model and Further Considerations

The core *Calyx of Held* stochastic model, i.e. the one featuring step-like or (one) wave-like Ca^{2+} uncaging, has been easily extended in order to support the presented experiments.

The generation of a second wave trivially consists of running a second identical wave-generator process (see Figure 3). Calling the second generator with a different name (`w2(1)`) allows us to more easily distinguish the two pulses, when needed. Moreover, in order to control the delay of the second wave, a process that activates the generator after a parametric nondeterministic delay has been introduced: `snd_w() = delay@125.0; w2(1)`. Although varying the delay parameter has allowed us to simulate a wide range of delays (some of which reported in Figure 4), this experiment has suggested the utility of a more precise time control over processes. In general, there might be several cases in which one would like to be able to express something like

```
run 1 of w(1) at 0.002
```

meaning that the stochastic process `w(1)` is supposed to start at *about* (in a sense to be specified according to the stochastic nature of the approach) time 0.002 of the simulation. This issue is under study.

The presence of residual calcium has been analogously modeled by modularly adding a generative process. Tuning the recursive branching factor and the delay parameter (e.g. `80` and `4000` for the case of `w(cnt)` in Figure 4) has allowed us to control the amount of Ca^{2+} and the time interval needed by the extrusion mechanism to clear it.

Besides the temporal aspects, also the spatial ones have needed to be engineered so that the desired behaviour is suitably modeled. According to the multi-pool hypothesis we have distinguished the reluctant vescicles of replenishing pools (`inf_v()`) from those undocked (`rct_v()`) and those docked (`rv()`):

```
inf_v() = !vinfgo; rct_v()

rct_v() = do !rvgo; rv()  or  !bvinfgo; inf_v()

rv()    = do !vca;  rv_ca()  or  !brvgo; rct_v()
```

Accordingly to the presented model, the behaviour of `rv()` is undistinguishable from that of the readily releasable `v()` with respect to calcium binding. Indeed, note that once docked, `rv()` interacts with `ca()` through the same channel `vca` used by `v()`. This preserves correctness of the model since, coherently with the interpretation of docked `rv()`, both kind of vescicles participate to calcium dynamics (and both concur, insisting on the same communication channels, to the Gillespie-based stochastic dynamics [21]). Such name distinction allows us to clearly mark fast and slow vescicles in Figure 5.

Although this reading of a "spatially" distributed (over different pools or classes of processes) behaviour has allowed us to perform coherent simulations, it seems worth investigating more expressive spatial primitives. A starting point is naturally represented by location-aware calculi *a là* Ambient calculus, like the Brane Calculi [5]. However, the problem of understanding the relation between stochastic dynamics and spatial information, e.g. how the traversal of an axon modifies signal strength, and hence its "interaction capability", is still open. Along this line goes the possibility of dynamically modifying the stochastic rates during a run, as advocated in Section 3.2.

4 Conclusion

The presented results are encouraging about the validity of the stochastic approach in studying the synaptic processes, which consist of many discrete-like events and involve arrays of vescicles and hundreds of different molecules. Many of these molecules have roles in the process of synaptic transmission which still are not fully understood [35]. We started by studying the release process by using data of a simplified experimental model, in which the concentration of Ca^{2+} was controlled and homogeneous. Under these hypotheses, the issue of spatial locality could not have been considered. We have extended the model in order to study events taking place during prolonged neural activity. To this aim, we added more details on vescicle trafficking and cluster compartimentalisation [20], and we studied short-term synaptic plasticity, facilitation and synaptic depression. Embedding these processes in the model might shed some light on the ways the nervous system processes and stores information. In order to support these, and others, developments the underlying process calculus might be extended, too. Surely, the problem of expressing locality, already addressed by several calculi, could be valuably addressed within a stochastic viewpoint.

References

1. Rediscovering Biology - Molecular to Global Perspectives - Neurobiology. http://www.learner.org/channel/courses/biology/support/10_neuro.pdf
2. Bollmann, J.H., Sakmann, B.: Control of synaptic strength and timing by the release-site Ca^{2+} signal. Nature Neuroscience 8, 426–434 (2005)
3. Bracciali, A., Brunelli, M., Cataldo, E., Degano, P. In: silico stochastic simulation of Ca^{2+} triggered synaptic release. In: Network Tools and Application in Biology (NETTAB07) (to appear, 2007)

4. Calder, M., Gilmore, S., Hillston, J.: Modelling the influence of rkip on the erk signalling pathway using the stochastic process algebra pepa. In: Priami, C., Ingólfsdóttir, A., Mishra, B., Nielson, H.R. (eds.) Transactions on Computational Systems Biology VII. LNCS (LNBI), vol. 4230, pp. 1–23. Springer, Heidelberg (2006)
5. Cardelli, L.: Brane calculi-interactions of biological membranes. In: Vincent, V., Schachter, V. (eds.) Proceedings of Computational Methods in Systems Biology, pp. 257–280. Springer, Heidelberg (2004)
6. Cassman, M., Arkin, A., Doyle, F., Katagiri, F., Lauffenburger, D., Stokes, C.: System Biology - International Research and Development. Springer, The Netherlands (2007)
7. Destexhe, A., Mainen, Z.F., Sejnowski, T.J.: Synthesis of models for excitable membrane, synaptic transmission and neuromodulation using a common kinetic formulation. The Journal of Computational Neuroscience 1, 195–231 (1994)
8. Fall, C.P., Marland, E.S., Wagner, J.M., Tyson, J.J. (eds.): Computational Cell Biology. Springer, New York (2002)
9. Felmy, F., Neher, E., Schneggenburger, R.: Probing the intracellular calcium sensitivity of transmitter release during synaptic facilitation. Neuron 37, 801–811 (2003)
10. Frotscher, M., Gundelfinger, E., Jonas, P., Neher, E., Seeburg, P.: The most important recent advances in synsapse research from my point of view - and what remains to be done. Cell Tissue Res. 326, 203–204 (2006)
11. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry 81, 2340–2361 (1977)
12. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
13. Kierzek, A.M.: Stocks: Stochastic kinetic simulations of biochemical system with gillespie algorithm. Bioinformatics 18, 470–481 (2002)
14. Kitano, H.: Systems biology: a brief overview. Science 295(5560), 1662–1664 (2002)
15. Koch, C., Segev, I. (eds.): Methods in Neuronal Modeling. The MIT Press, Cambridge (1998)
16. Lecca, P., Priami, C., Quaglia, P., Rossi, B., Laudanna, C., Costantin, G.: A stochastic process algebra approach to simulation of autoreactive lymphocyte recruitment. SIMULATION: Trans. of the society for modelling and simulation international 80(4), 273–288 (2004)
17. Meinrenken, C.J., Borst, J.G.G., Sakmann, B.: The Hodgkin-Huxley-Katz Prize Lecture. Local routes revisited: the space and time dependence of the Ca signal for phasic transmitter release at the rat calyx of Held. (In press)
18. Milner, R.: Communicating and Mobile Systems: The π-Calculus. Cambridge University Press, Cambridge (1999)
19. Nagasaki, M., Onami, S., Miyano, S., Kitano, H.: Bio-calculus: its concept and molecular interaction. Genome Informatics 10, 133–143 (1999)
20. Neher, E.: A comparison between exocytic control mechanisms in adrenal chromaffin cells and a glutamatergic synapse. Pflugers. Arch. - Eur. J. Physiol. 453, 261–268 (2006)
21. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. In: Proceedings of Bioconcur'04, ENTCS, Elsevier, Amsterdam (2004)
22. Priami, C.: Stochastic π-calculus. The Computer Journal 36(6), 578–589 (2004)
23. Priami, C., Regev, A., Shapiro, E., Silvermann, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Theoretical Computer Science 325(1), 141–167 (2004)

24. Regev, A., Panina, E., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartements. Theoretical Computer Science 325(1), 141–167 (2004)
25. Regev, A., Shapiro, E.: Cellular abstractions: Cells as computation. Nature 419, 343 (2002)
26. Sakaba, T.: Roles of fast-releasing and the slowly releasing vescicles in synaptic transmission at the calyx of Held. Journal if Neuroscience 26, 5863–5871 (2006)
27. Sakaba, T., Schneggenburger, R., Neher, E.: Estimation of quantal parameters at the calyx of Held synapse. Neuroscience Research 44, 343–356 (2002)
28. Sakaba, T., Stein, A., Jahn, R., Neher, E.: Distinct kinetic changes in neurotransmitter release after snare protein cleavage. Science 309, 491–494 (2005)
29. Schneggenburger, R., Forsythe, I.D.: The calxy of Held. Cell Tissue Res. 326, 311–337 (2006)
30. Schneggenburger, R., Neher, E.: Intracellular calcium dependence of transmitter release rates at a fast central syanpse. Nature 46, 889–893 (2000)
31. Schneggenburger, R., Neher, E.: Presynaptic calcium and control of vescicle fusion. Current Opinion in Neurobiology 15, 266–274 (2005)
32. Schneggenburger, R., Sakaba, T., Neher, E.: Vescicle pools and short-term synaptic depression: lessons from a large synapse. TINS 25, 206–212 (2002)
33. Smolen, P.D., Baxter, D.A., Byrne, J.H.: Mathematical modeling and analysis of intracellular signaling pathways. In: Byrne, J.H., Roberts, J.L. (eds.) From Molecules to Networks - An Introduction to Cellular and Molecular Neuroscience, pp. 391–429. Elsevier - Academic Press, Amsterdam (2004)
34. Sorensen, J.B.: Formation, stabilization and fusion of the readily releasable pool of secretory vescicles. Pflugers Arc - European Journal of Neuroscience 448, 347–362 (2004)
35. Sudhof, T.C.: The synaptic vescicle cycle. Annual Review of Neuroscience 27, 509–547 (2004)
36. Weis, S., Schneggenburger, R., Neher, E.: Properties of a model of ca^{2+} -dependent vescicle pool dynamics and short term synaptic depression. Biophysical Journal 77, 2418–2429 (1999)
37. Wilkinson, D.J.: Stochastic Modelling for System Biology. Chapman and Hall - CRC, London (2006)
38. Zucker, R.S., Kullmann, D.M., Schwartz, T.L.: Release of neurotransmitters. In: Byrne, J.H., Roberts, J.L. (eds.) From Molecules to Networks - An Introduction to Cellular and Molecular Neuroscience, pp. 197–244. Elsevier - Academic Press, Amsterdam (2004)

Modelization and Simulation of Nano Devices in nanoκ Calculus

Alberto Credi[1], Marco Garavelli[1], Cosimo Laneve[2], Sylvain Pradalier[3], Serena Silvi[1], and Gianluigi Zavattaro[2]

[1] Dipartimento di Chimica "G. Ciamician", Università di Bologna
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna
[3] Ecole Polytechnique, Paris

Abstract. We develop a process calculus – the nanoκ Calculus – for modeling, analyzing and predicting the properties of molecular devices. The nanoκ Calculus is equipped with a simple stochastic model, that we use to model and simulate the behaviour of a molecular shuttle, a basic nano device currently used for building more complex systems.

1 Introduction

In 2006 the University of Bologna funded an interdisciplinary project of its Departments of Chemistry and Computer Science – the *CompReNDe Project* (Compositional and executable Representations of Nano Devices). The project combines the expertises of two groups, one specialized in the design and construction of devices and machines of molecular size [3,2] and the other one qualified in formal models, based on the theory of process calculi, for describing and analyzing molecular systems [7,14]. Such expertises are joined together in order to accomplish three main endeavours: (i) deliver a programming model for describing molecular machines that is also amenable to automated simulations and verifications by means of existing algorithms, (ii) apply the model for a formal analysis of real cases of molecular machines to possibly reveal complex behaviours that have not been experimentally observed yet, and (iii) use the simulations as tools to assist chemists in the design of novel nano devices.

The CompReNDe research activity started with the initial goal of formalizing a *[2]rotaxane* [20] into the κ calculus [7] in order to simulate its behaviour *in silico* by means of some contemporary stochastic evaluator [10,19,5]. [2]rotaxanes [20] (simply rotaxanes in the following) are systems composed of a molecular axle surrounded by a ring-type (macrocyclic) molecule. Bulky chemical moieties ("stoppers") are placed at the extremities of the axle to prevent the disassembly of the system. In rotaxanes containing two different recognition sites on the axle ("stations"), it is possible to switch the position of the macrocyclic ring between the two stations by an external energy input as illustrated in Figure 1. Several rotaxanes of this kind, known as *molecular shuttles*, have been already developed (see [6] and the references therein) and used for building more complex systems [13,12,2].

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 168–183, 2007.

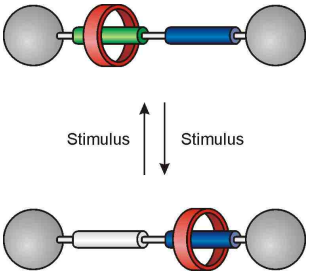


Fig. 1. Schematic representation of a two-station rotaxane and its operation as a controllable molecular shuttle

The κ calculus is a formal language idealizing protein-protein interactions, as a particular restricted kind of graph-rewriting. Bindings are explicit: proteins are nodes with fixed numbers of sites, complexes are connected graphs built over such nodes where bonds are represented by *names*. Biological reactions are modeled by two kinds of rewriting rules: *complexations*, which create bonds, and *decomplexations*, which destroy bonds. Notably, the κ calculus has been compiled into π-calculus [16] introducing a finer-grained concurrent model, the mκ calculus, where reactions have to be at most binary. The significant property of mκ calculus is to be *protein-centred*, rather then reaction-centred as it was the case for κ calculus, thus being amenable to distributed implementations.

We therefore undertook the formalization of a molecular shuttle in mκ calculus and we soon realized that such calculus was inadequate as well. The mκ calculus is much too verbose because it compels designers to reason in terms of bonds and complexations and decomplexations. There are reactions that are neither complexations nor decomplexations, such as the *ion exchanges*. These reactions, used in our molecular shuttle to stimulate the movement of the macrocyclic ring, might be implemented by sequences of complexations and decomplexations, thus changing the granularity of the chemical semantics. The mκ calculus model is much too abstract because it overlooks quantitative aspects. Such aspects, in particular reaction rates and the derived stochastic semantics are a must for providing meaningful simulations of molecular machines.

We overcome these inadequacies of the mκ calculus by defining a new model, the nanoκ *Calculus*, having three types of reactions – *creations*, *destructions*, and *exchanges* – and retaining a stochastic semantics. This stochastic semantics is problematic for the nanoκ Calculus because it uses names for representing molecular bonds. In this respect, our model is close to Milner's pi calculus [16]. However, instead of following the techniques of the stochastic pi calculus [18], we have preferred for nanoκ Calculus to extend Cardelli's language of stochastic interacting processes [4]. In facts, in this way, we get a simple model that may be easily simulated or verified by means of existing well known algorithms [9].

We then apply the nanoκ Calculus to describe and analyze an instance of rotaxane, RaH [15,1], for which the dynamic behaviour has been experimentally

characterized in detail [8]. We have considered two groups of simulations. The first ones are used to validate the model, checking whether the experiments reproduced *in silico* coincides with those already performed *in vitro*. The second ones simulate *in silico* the expected behaviour of the rotaxane RaH under conditions not yet observed *in vitro*. Interestingly, we show that under extreme conditions of very low concentration of rotaxane RaH, some of the assumptions, usually taken about the behaviour of the rotaxane in standard conditions of concentration, are no longer valid.

2 The nanoκ Calculus: Syntax and Semantics

Two disjoint countable sets of names will be used: a set of *species*, ranged over by A, B, C, $\cdots$; and a totally ordered set of *bonds*, ranged over by x, y, z, $\cdots$. Species are sorted according to the number of *fields* and *sites* they possess. Let $\mathfrak{s}_f(\cdot)$ and $\mathfrak{s}_s(\cdot)$ be two functions returning naturals; the integers 1, 2, $\cdots$, $\mathfrak{s}_f(A)$ and 1, 2, $\cdots$ $\mathfrak{s}_s(A)$ are respectively the fields and the sites of A. ($\mathfrak{s}_f(A) = 0$ means there is no field; $\mathfrak{s}_s(A) = 0$ means there is no site). In the following, fields are ranged over by h, i, j, $\cdots$; sites are ranged over by a, b, c, $\cdots$.

Sites may be either *bound* to other sites or *unbound*, i.e. not connected to other sites. The state of sites are defined by injective maps, called *interfaces* and ranged over by σ, ρ, $\cdots$. Given a species A, its interfaces are partial functions from $\{1, \cdots, \mathfrak{s}_s(A)\}$ to the set of bonds or a special empty value ε. A site a is bound with bond x in σ if $\sigma(a) = x$; it is unbound if $\sigma(a) = \varepsilon$. For instance, if A is a species with three sites, $(2 \mapsto x, 3 \mapsto \varepsilon)$ is one of its interfaces. In order to ease the reading, we write this map as $2^x + 3$ (the empty value is always omitted). This interface σ does not define the state of the site 1, which may be bound or not. In the following, when we write $\sigma + \sigma'$ we assume that the domains of σ and σ' are disjoint. Interfaces, being injective on bonds, cannot express that the endpoints of a bond belong to the same species (*cf. self complexation* in [7]). This design choice simplifies the presentation of nanoκ Calculus.

Fields represent the internal state of a species. The values of fields are defined by maps, called *evaluations*, and ranged over by u, v, $\cdots$. For instance, if A is a species with three fields, $[1 \mapsto 5, 2 \mapsto 0, 3 \mapsto 4]$ is an evaluation of its. As before, we write this map as $1^5 + 2^0 + 3^4$. We assume there are finitely many internal states, that is every field h is mapped into values in $\{0, \cdots, n_h\}$. In the following, we use partial evaluations and, when we write the union of evaluations $u + v$, we implicitly assume that the domains of u and v are disjoint.

Definition 1 (Molecules and Solutions). *A* molecule $A[u](\sigma)$ *is a term where* u *is a total map on the fields of* A. Solutions, *ranged over by* S, T, $\cdots$, *are defined by the following grammar*

$$\mathsf{S} \quad ::= \quad A[u](\sigma) \;\mid\; \mathsf{S},\mathsf{S}$$

The operator "$,$*" is assumed to be associative, so* $(\mathsf{S},\mathsf{S}'),\mathsf{S}''$ *is equal to* $\mathsf{S},(\mathsf{S}',\mathsf{S}'')$ *(and we always omit parentheses).*

Solutions retain the property that bond names always occurs exactly twice. Let $\emptyset$ *be the empty map. We use the following shorthand notations:* $A(\sigma)$ *instead of* $A[\emptyset](\sigma)$*,* $A[u]$ *instead of* $A[u](\emptyset)$*, and simply* A *instead of* $A[\emptyset](\emptyset)$*.*

Example 1. As a running example we consider two typical chemical reactions:

- $Na + Cl \longleftrightarrow Na^+ + Cl^-$ (*sodium chloride*) and
- $H + H \longleftrightarrow H_2$ (*hydrogen gas*) .

In the first reaction, an ion is exchanged between two instances of species Na and Cl. The molecules of the two species can be in two possible states: either they have the extra ion Na^+ and miss an ion Cl^- or they are in the states with all the ions Na and Cl. We model these two possible states using one field *ion* with values 0 and 1 respectively denoting the absence or the presence of the ion. Formally we can use $Na[ion^0]$ and $Na[ion^1]$ for Na and Na^+, and $Cl[ion^0]$ and $Cl[ion^1]$ for Cl^- and Cl, respectively.

The second chemical reaction represents the creation/destruction of a bond between two hydrogen atoms. This may be described by using a site 1 and bond names. For instance, the solution with H_2 is modelled by $H(1^x), H(1^x)$. An unbound instance of hydrogen is simply represented by H, as its evaluation and interface are both empty.)

Definition 2 (Reactions). *Reactions of* nanoκ *Calculus are either* creations C*, or* destructions D*, or* exchanges E*. The format of the first two types is* $((A, a, u, u', \sigma), (B, b, v, v', \phi), \lambda)$*;while the format of exchanges is* $((A, u, u', \rho, \rho'), (B, v, v', \psi, \psi'), \lambda)$*, such that:*

1. $\mathrm{dom}(u') = \mathrm{dom}(u)$ *and* u *and* u' *are partial evaluations of* A*,* $\mathrm{dom}(v') = \mathrm{dom}(v)$ *and* v *and* v' *are partial evaluations of* B*,*
2. $\mathrm{ran}(\sigma) = \mathrm{ran}(\phi)$ *and* σ *and* ϕ *are interfaces of* A *and* B*, respectively, such that* $a \notin \mathrm{dom}(\sigma)$ *and* $b \notin \mathrm{dom}(\phi)$*;*
3. *(for exchanges)* ρ, ρ' *and* ψ, ψ' *are interfaces of* A *and* B*, respectively, with* $\mathrm{ran}(\rho') = \mathrm{ran}(\psi)$ *and either* $\rho = \rho'$ *and* $\psi = \psi'$ *or* $\rho = a^x + \rho''$*,* $\rho' = a + \rho''$ *and* $\psi = b + \psi''$*,* $\psi' = b^x + \psi''$*;*
4. *and* $\lambda \in \mathbb{R}^+ \cup \{\infty\}$*.*

For readability's sake, we write creations as $A[u](a+\sigma), B[v](b+\phi) \xrightarrow{\lambda} A[u'](a^x+\sigma), B[v'](b^x+\phi)$*, destructions as* $A[u](a^x+\sigma), B[v](b^x+\phi) \xrightarrow{\lambda} A[u'](a+\sigma), B[v'](b+\phi)$*, and exchanges as* $A[u](\rho), B[v](\psi) \xrightarrow{\lambda} A[u'](\rho'), B[v'](\psi')$*.*

The difference between the three kinds of rules is concerned with the modification of the interfaces: *creations* produce a new bond between the two unbound sites a and b, *destructions* remove the bond between the sites a and b, while *exchanges* either leave the interfaces unchanged or move one bond from a reactant to the other (bond-flipping exchange).[1]

[1] The terms creation and destruction have been preferred to *complexation* and *decomplexation* used in [7,14] because they have a more neutral chemical meaning.

Example 2. The nanoκ Calculus reactions that corresponds to the two reactions of the sodium chloride are

$$Na[ion^0], Cl[ion^1] \overset{100}{\rightarrow} Na[ion^1], Cl[ion^0]$$
$$Na[ion^1], Cl[ion^0] \overset{10}{\rightarrow} Na[ion^0], Cl[ion^1]$$

where we have considered a rate 100 for the left to right direction and 10 for the right to left direction.

The nanoκ Calculus reactions that corresponds to the two reactions of the hydrogen gas are

$$H(1), H(1) \overset{5}{\rightarrow} H(1^x), H(1^x) \qquad H(1^x), H(1^x) \overset{0.05}{\rightarrow} H(1), H(1)$$

where the right direction has been given rate 5 and the left direction has been given rate 0.05.

The formal definition of *reactants* and the corresponding *products* of reactions follows. We use μ to range over $\rho_L, \imath, x$ and $\rho_R, \imath, x$ and $\rho_L, \imath$ and $\rho_R, \imath$ and ρ. Let $\overline{\mu}$ be the following operation (notice that $\overline{\overline{\mu}} = \mu$):

$$\overline{\mu} \stackrel{def}{=} \begin{cases} \rho_R, \imath, x & \text{if } \mu = \rho_L, \imath, x \\ \rho_L, \imath, x & \text{if } \mu = \rho_R, \imath, x \\ \rho_R, \imath & \text{if } \mu = \rho_L, \imath \\ \rho_L, \imath & \text{if } \mu = \rho_R, \imath \\ \rho & \text{if } \mu = \rho \end{cases}$$

Definition 3 (Basic transition relation). *The basic transition relation of solutions, written* $\xrightarrow{\mu}_\ell \cup \xrightarrow{\mu}_{\ell,\ell'}$*, is the least relation that satisfies the following rules (*$\imath$ *are always injective renamings on bonds):*

- *(creations) if* $\rho = A[u](a+\sigma), B[v](b+\phi) \overset{\lambda}{\rightarrow} A[u'](a^x+\sigma), B[v'](b^x+\phi)$ *and* $\mathrm{dom}(\imath) = \mathrm{ran}(\sigma)$ $(= \mathrm{ran}(\phi))$ *and* $z \notin \mathrm{ran}(\sigma \circ \imath + \nu)$ *then both* $A[u+w](a + \sigma \circ \imath + \nu) \xrightarrow{\rho_L,\imath,z}_1 A[u'+w](a^z + \sigma \circ \imath + \nu)$ *and* $B[v+w](b + \phi \circ \imath + \nu) \xrightarrow{\rho_R,\imath,z}_1 B[v'+w](b^z + \phi \circ \imath + \nu)$*;*
- *(destructions) if* $\rho = A[u](a^x+\sigma), B[v](b^x+\phi) \overset{\lambda}{\rightarrow} A[u'](a+\sigma), B[v'](b+\phi)$ *and* $\mathrm{dom}(\imath) = \mathrm{ran}(\sigma)$ $(= \mathrm{ran}(\phi))$ *then both* $A[u+w](a^x + \sigma \circ \imath + \nu) \xrightarrow{\rho_L,\imath,x}_1 A[u'+w](a+\sigma\circ\imath+\nu)$ *and* $B[v+w](b^x+\phi\circ\imath+\nu) \xrightarrow{\rho_R,\imath,x}_1 B[v'+w](b+\phi\circ\imath+\nu)$*;*
- *(exchanges) if* $\rho = A[u](\sigma), B[v](\phi) \overset{\lambda}{\rightarrow} A[u'](\sigma'), B[v'](\phi')$ *and* $\mathrm{dom}(\imath) = \mathrm{ran}(\sigma)$ $(= \mathrm{ran}(\phi))$ *then both* $A[u+w](\sigma \circ \imath + \nu) \xrightarrow{\rho_L,\imath,\upsilon}_1 A[u'+w](\sigma' \circ \imath + \nu)$ *and* $B[v+w](\phi \circ \imath + \nu) \xrightarrow{\rho_R,\imath,\upsilon}_1 B[v'+w](\phi' \circ \imath + \nu)$*, where* υ *is either* ε *or* $\imath(x)$*, according to* $\mathrm{ran}(\sigma) \setminus \mathrm{ran}(\sigma')$ *is* $\emptyset$ *or* $\{x\}$*;*
- *(lifts) if* $\mathsf{S} \xrightarrow{\mu}_\ell \mathsf{S}'$ *and, when* ρ *is a creation,* $(\mathrm{name}(\mathsf{S}') \setminus \mathrm{name}(\mathsf{S})) \cap \mathrm{name}(\mathsf{T}) = \emptyset$*, then both* $\mathsf{S},\mathsf{T} \xrightarrow{\mu}_\ell \mathsf{S}',\mathsf{T}$ *and* $\mathsf{T},\mathsf{S} \xrightarrow{\mu}_{\ell'+\ell} \mathsf{T},\mathsf{S}'$*, where* T *has* ℓ' *molecules;*
- *(communications) if* $\mathsf{S} \xrightarrow{\mu}_\ell \mathsf{S}'$ *and* $\mathsf{T} \xrightarrow{\overline{\mu}}_{\ell'} \mathsf{T}'$ *then* $\mathsf{S},\mathsf{T} \xrightarrow{\rho}_{\ell,\ell''+\ell'} \mathsf{S}',\mathsf{T}'$*, where* ρ *is the rule of* μ *and* S *has* ℓ'' *molecules. If* ρ *is a creation, then the bond used by the reaction is the least one that is not used in* S,T*.*

The basic transition relation definitely deserves to be spelled out. A reaction, such as $Na[ion^0], Cl[ion^1] \stackrel{100}{\Rightarrow} Na[ion^1], Cl[ion^0]$ is a *schema*, namely it only addresses the fields and the the sites of the reactants that are useful for the reaction. For example, it may be the case that Na retains a site to be used for other complexes, such as the *sodium peroxide*. In this case, the rule may be applied either to $Na[ion^0]$, where the site is unbound, or to $Na[ion^0](1^x)$. In this latter case, the reaction is instantiated as the transition:

$$Na[ion^0](1^x), Cl[ion^1] \stackrel{\rho}{\longrightarrow}_{1,2} Na[ion^1](1^x), Cl[ion^0]$$

The basic transition relation is indexed by numbers. Since the solutions are sequences, these numbers give the exact positions of the reactants in the sequences. In the first three cases, the position is always 1 because the solution consists of one molecule. In the fourth case, the index is increased by the number of the molucules on the left, if any. The last case models a reaction: the solution is split into two parts S and T containing the reactants at positions ℓ and ℓ', respectively. In the composite solution S,T, the reactants are at ℓ and $\ell'' + \ell'$, where ℓ'' is the number of molecules of S. For example let $k\mathsf{M}$ be $\underbrace{\mathsf{M}, \cdots, \mathsf{M}}_{k \text{ times}}$ and let ρ be the hydrogen gas reaction. The following transitions are possible

$$\begin{aligned} 3H(1) &\stackrel{\rho}{\longrightarrow}_{1,2} 2H(1^x), H(1) \\ 3H(1) &\stackrel{\rho}{\longrightarrow}_{1,3} H(1^x), H(1), H(1^x) \\ 3H(1) &\stackrel{\rho}{\longrightarrow}_{2,3} H(1), 2H(1^x) \end{aligned}$$

The basic transition relation is labelled by finite injective renamings. To clarify this point, consider the creation $\varrho = Na(1^x + 2), Na(1^x + 2) \stackrel{10}{\Rightarrow} Na(1^x + 2^y), Na(1^x + 2^y)$ (a bond is created between two sodium molecules provided they are already bound). Then take the solution $Na[ion^0](1^z + 2), Na[ion^0](1^v + 2), Na[ion^1](1^z + 2), Na[ion^0](1^v + 2)$. We derive the expected transition

$$\begin{aligned} &Na[ion^0](1^z + 2), Na[ion^0](1^v + 2), Na[ion^1](1^z + 2), Na[ion^0](1^v + 2) \\ &\stackrel{\varrho}{\longrightarrow}_{1,3} Na[ion^0](1^z + 2^y), Na[ion^0](1^v + 2), Na[ion^1](1^z + 2^y), Na[ion^0](1^v + 2) \end{aligned}$$

following a structured operational semantics approach [17]. Namely, we focus on the single reactants and lift the transitions to ",”-contextes. This is correct inasmuch as one records the instantiation of bonds in the left-hand sides of reactions with the actual names of the molecules: the two reactants must instantiate bonds in the same way. This is the reason why the first two molecules of the above solution cannot react with ϱ. More precisely, $Na[ion^0](1^z + 2) \stackrel{\varrho_{L,\imath,y}}{\longrightarrow}_1 Na[ion^0](1^z + 2^y)$, where $\imath = [x \mapsto z]$, and $Na[ion^0](1^v + 2) \stackrel{\varrho_{R,\imath,y}}{\not\longrightarrow}_1$.

Our final remarks regard the fourth and fifth items of Definition 3. Whenever $\mathsf{S} \stackrel{\rho,\imath,x}{\longrightarrow}_\ell \mathsf{T}$ and ρ is a creation, the basic transition relation also admits $\mathsf{S} \stackrel{\rho,\imath,y}{\longrightarrow}_\ell \mathsf{T}\{{}^{y}/_{x}\}$, where y is fresh. This nondeterminism is removed when the reaction occurs because the bond has to be the least name not occurring in S. It is also worth to notice that there is no rule lifting a transition $\stackrel{\mu}{\longrightarrow}_{\ell,\ell'}$ to a context ",":

we use the associativity of , to partition a solution S into S',S'' such that the reactants are in S' and S''.

The basic transition relation is excessively intensional to be sensible for chemistry. Consider a solution containing hundreds of molecules of the species A and B that could react with ρ. The relation $\xrightarrow{\mu}_{\ell,\ell'}$ distinguishes the two pairs of reactants, and this is not possible in practice. More reasonably, the transition relation should represent *collectively* all the possible combinations of one molecule of species A with one molecule of species B. For instance, the solution A,A,B transits with $\xrightarrow{\rho}_{1,3}$ and $\xrightarrow{\rho}_{2,3}$. Abstracting out the order of the molecules, we obtain a unique transition whose rate is twice the rate of ρ. However quotienting the solutions with commutativity axioms of "," does not yield an adequate extensionality. In facts, when ρ is a destruction, between A and B, the solution $A(a^x),A(a^y),B(a^x),B(a^y)$ transits with $\xrightarrow{\rho}_{1,3}$ and $\xrightarrow{\rho}_{2,4}$ into two solutions that cannot be equated by permutations of the molecules in the solution. In these cases one has to use injective renamings of bonds.

Definition 4 (Structural equivalence). *The structural equivalence between solutions, noted $\equiv$, is the least equivalence satisfying the following two rules (we remind that solutions are already quotiented by associativity of ","):*

1. $\mathsf{S},\mathsf{T} \equiv \mathsf{T},\mathsf{S}$;
2. $\mathsf{S} \equiv \mathsf{T}$ *if there exists an injective renaming $\imath$ on bonds such that* $\mathsf{S} = \imath(\mathsf{T})$.

Example 3. Commutativity and injective renaming of the structural equivalence permit to prove the two following equivalences, respectively

$$Na[h^0],Cl[h^1] \;\equiv\; Cl[h^1],Na[h^0] \qquad H(b^x),H(b^x) \;\equiv\; H(b^y),H(b^y)$$

Combining both commutativity and injective renaming we can prove that

$$H(b^x),H(b^x),H(b^z),H(b^z) \;\equiv\; H(b^y),H(b^k),H(b^k),H(b^y)$$

Proposition 1. *Let $\mathsf{S} \equiv \mathsf{S}'$.*

1. *If $\mathsf{S} \xrightarrow{\mu}_{\ell} \mathsf{T}$ then there is T' and a renaming $\imath$ such that $\mathsf{S}' \xrightarrow{\imath(\mu)}_{\ell'} \mathsf{T}'$ and $\mathsf{T}' \equiv \mathsf{S}'$;*
2. *if $\mathsf{S} \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}$ then there is T' such that $\mathsf{S}' \xrightarrow{\rho}_{\ell'',\ell'''} \mathsf{T}'$ and $\mathsf{T}' \equiv \mathsf{S}'$.*

The following notations are relevant for the definition of the stochastic transition relation:

- $rate(\rho)$ returns the rate of the reaction ρ;
- $next(\mathsf{S}) = \{(\rho_{\ell,\ell'},\mathsf{T}) \mid \mathsf{S} \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}\}$; $next_\infty(\mathsf{S}) = \{(\rho_{\ell,\ell'},\mathsf{T}) \mid \mathsf{S} \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}$ and $rate(\rho) = \infty\}$;
- $\mathcal{S}$ *has finite rates* if, for every $(\rho_{\ell,\ell'},\mathsf{T}) \in \mathcal{S}$, $rate(\rho)$ is not ∞;
- let $\mathcal{S}$ be a set of pairs (X,T') (the second element is a solution; the first one is not specified), $[\mathcal{S}]_\mathsf{T}$ is the subset of $\mathcal{S}$ of those pairs (X,T') such that $\mathsf{T}' \equiv \mathsf{T}$;

– $can(\mathcal{S})$ is defined over sets of pairs (X, T) such that the solutions occurring as second element of the pairs are all structurally equivalent. It returns a solution S such that there is X with $(X, \mathcal{S}) \in \mathsf{S}$.

Definition 5 (Stochastic transition relation). *The* nanoκ *Calculus stochastic transition relation* $\stackrel{\lambda}{\longmapsto}$*, where* $\lambda \in \mathbb{R}^+ \cup \{\infty\}$*, is the least relation satisfying the following rules:*

– *if* $\mathsf{S} \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}$ *and* $rate(\rho) = \infty$ *then* $\mathsf{S} \stackrel{\infty}{\longmapsto} can([next_\infty(\mathsf{S})]_\mathsf{T})$*;*
– *if* $\mathsf{S} \xrightarrow{\rho}_{\ell,\ell'} \mathsf{T}$ *and* $next(\mathsf{S})$ *has finite rates then* $\mathsf{S} \stackrel{\lambda}{\longmapsto} can([next(\mathsf{S})]_\mathsf{T})$*, where*

$$\lambda \;=\; \sum_{(\rho_{\ell,\ell'},\mathsf{T}') \in [next(\mathsf{S})]_\mathsf{T}} rate(\rho)$$

We notice that, by definition, the nanoκ Calculus stochastic transition system is such that there is no state with outgoing $\stackrel{\infty}{\longmapsto}$ *and* $\stackrel{\lambda}{\longmapsto}$ (λ finite) transitions. Hereafter, the states with $\stackrel{\infty}{\longmapsto}$ outgoing transitions are called *transient states*, the other ones are called *markovian states*.

The interrelation between basic and stochastic transition relations is as follows: the stochastic one partitions the products of a solution (according to the basic transition relation) into equivalence classes, takes a canonical representative of the class, and defines a transition whose label is the sum of the rates of the reactions in the basic one that yield solutions in the equivalence class.

Example 4. As examples of stochastic transitions, we consider the reactions of sodium chloride (called ρ) and hydrogen gas (called ρ') of Example 1 for the solution $2Na[ion^0], Cl[ion^1], 3H$. This solution may transit with $\xrightarrow{\rho}_{1,3}$ and $\xrightarrow{\rho}_{2,3}$ into solutions that are structural equivalent. Therefore we obtain a unique stochastic transition:

$$2Na[ion^0], Cl[ion^1], 3H(1) \stackrel{200}{\longmapsto} Na[ion^1], Na[ion^0], Cl[ion^1], 3H(1)$$

We also observe that there is a unique transition $\stackrel{15}{\longmapsto}$ outgoing the initial solution and corresponding to $\xrightarrow{\rho'}_{4,5}$, $\xrightarrow{\rho'}_{4,6}$, and $\xrightarrow{\rho'}_{5,6}$.

3 Markov Chains and the nanoκ Calculus

The stochastic transition relation of nanoκ Calculus corresponds to an *Interactive Markov Chain (IMC) transition system* with only *silent interactive transitions* [11]. These transitions, which are those labelled ∞ in our model, are executed in the IMC model instantaneously and under the *maximal progress assumption*. That is, the so-called *sojourn time* in a transient state is 0, which amounts to favour silent interactive transitions to those labelled with finite rates (called *markovian* transitions). On the contrary, in a markovian state with n outgoing markovian transitions labelled $\lambda_1, \cdots, \lambda_n$, the probability that the

sojourn time is less than t is exponentially distributed with rate $\sum_i \lambda_i$, i.e. $\mathsf{Prob}\{delay < t\} = 1 - e^{-t\sum_i \lambda_i}$, and the probability that the j-th transition is taken is $\lambda_j/(\sum_i \lambda_i)$.

However the models underlying traditional simulation algorithms such as [9] are *Continuos Time Markov transition systems (CTMC)* that do not include interactive transitions. Having a CTMC is therefore primary to run automatic analysis tools for experimenting *in silico* the dynamics of nano-machines specifications in nanoκ Calculus.

The mismatch between IMC with only silent actions and CTMC systems is due to two main reasons: (i) the nondeterminism and (ii) the persistency of the silent interactive transitions. As regards (i), consider two silent actions that apply to the same reactants and give two different products. If these products have only markovian transitions it is not possible to collect them in a unique solution. As regards (ii), if an infinite sequence of silent interactive transitions exists then the simulation time of the CTMC system will not advance anymore. Therefore collapsing all these transitions, by identifying the initial and final solutions of the sequence, is again not possible.

However, there are cases where the downgrading of an IMC system to a CTMC one is possible without modifying the semantics. This is *when all silent actions may be partitioned into confluent directed acyclic graphs of finite depth*. In fact, when the silent interactive transitions are partitioned into confluent directed acyclic graphs, there are no loops (there is no infinite sequence of silent interactive transitions), and all sequences of silent interactive transitions starting from the same state share the same final state, to which the initial state may be safely collapsed. The meaning of this collapse is that we are removing a finite amount of work which is performed in zero time.

The formal definition of downgrading of IMC to CTMC systems follows. We first introduce the auxiliary function *next markovian state* defined on solutions and yielding sets:

– $nextm(\mathsf{S}) = \{((\lambda, \mathsf{T}'), \mathsf{T}) \mid \mathsf{S} \xmapsto{\lambda} \mathsf{T}' \xmapsto{\infty}{}^{*} \mathsf{T}$ and $\lambda \in \mathbb{R}^+$ and $\mathsf{T} \not\xmapsto{\infty}\}$

We notice that $nextm(\mathsf{S})$ is undefined when S is transient.

Definition 6 (Downgrading of an IMC system). *An IMC system* $(\mathcal{S}, \xmapsto{\lambda})$ *is* strictly-markovian *if*

1. *states are either transient or markovian and*
2. *every subsystem consisting of silent interactive transitions is a confluent direct acyclic graph of finite depth.*

Let $(\mathcal{S}, \xmapsto{\lambda})$ *be strictly-markovian; the transition relation* $\overset{\nu}{\Longmapsto}$, *where* $\nu \in \mathbb{R}^+$, *is the least one such that:*

– *if* S *is markovian then* $\mathsf{S} \overset{\nu}{\Longmapsto} can([nextm(\mathsf{S})]_{\mathsf{T}})$ *with*

$$\nu = \sum_{((\lambda,\mathsf{T}'),\mathsf{T}'') \in [nextm(\mathsf{S})]_{\mathsf{T}}} \lambda$$

It is easy to verify that the relation $\stackrel{\nu}{\Longmapsto}$ defines a CTMC system. Moreover, the properties below are direct consequences of the construction:

- the probability distribution of the sojourn time in a markovian state is the same in the IMC and in the downgraded CTMC and
- the probability that one of the paths $\mathsf{S} \stackrel{\lambda}{\longmapsto} \stackrel{\infty}{\longmapsto}^{*} \mathsf{T}'$ with $\mathsf{T}' \equiv \mathsf{T}$ is taken in the IMC corresponds to the probability the unique transition $\mathsf{S} \stackrel{\lambda'}{\Longmapsto} \mathsf{T}''$, with $\mathsf{T}'' \equiv \mathsf{T}$, is taken in the downgraded CTMC.

Actually the correspondence between strictly-markovian IMC and the associated CTMC is much stronger: the IMC semantics, the *markovian bisimulation* [11], is still a markovian bisimulation on the CTMC when restricted to its states. We will detail the correspondence in the full paper.

4 nanoκ Calculus at Work: The Rotaxane Case Study

The investigated rotaxane RaH (Figure 2) [15,1] is made of a stoppered axle containing an ammonium (A) and an electron acceptor bipyridinium (B) stations that can establish hydrogen-bonding and charge-transfer interactions, respectively, with the ring component, which is a crown ether with electron donor properties. Since the hydrogen bonding interactions between the macrocyclic ring and the ammonium center are much stronger than the charge-transfer interactions of the ring with the bipyridinium unit, the rotaxane exists as only one of the two possible translational isomers, denoted as RaH in Figure 2. In solution, addition of a base (e.g., tributylamine) converts the ammonium center into an amine function, giving the transient state Ra that is transformed into the stable state Rb as a consequence of the displacement of the macrocycle

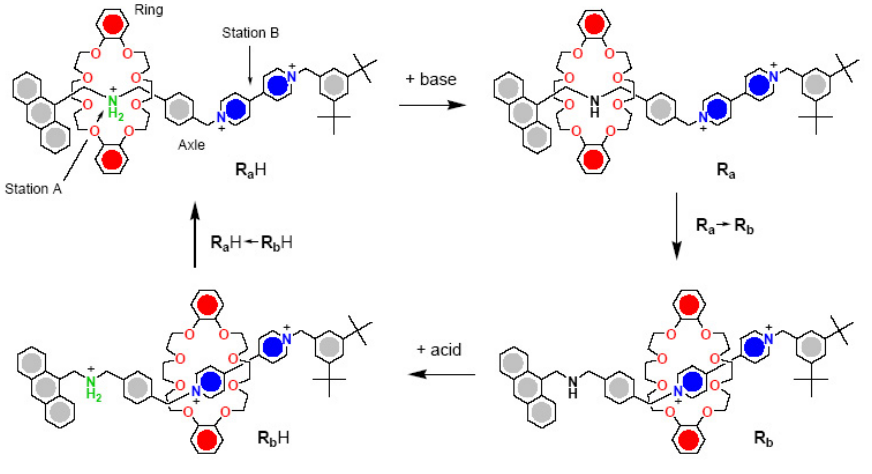


Fig. 2. Schematic representation of the shuttling processes of the molecular ring in the examined rotaxane

onto the B station. The process can be reversed by addition of acid (e.g., trifluoroacetic acid) and the initial state is restored, passing through the transient state denoted as RbH. Nuclear magnetic resonance, absorption and luminescence spectroscopic experiments, together with electrochemical measurements, indicate that the acid-base controlled switching, which is fully reversible and relatively fast, exhibits a clear-cut on-off behaviour [1].

The Rotaxane RaH is particularly appropriate to test the modeling approach described in the present paper because it is one of the very few cases wherein not only the thermodynamic properties, but also the dynamic behavior of the system have been experimentally characterized in detail. Specifically, the macrocycle's shuttling process between the ammonium/amine and bipyridinium stations in this rotaxane, driven by the successive addition of base and acid, have been investigated in solution [8]. The rate constants for the "forward" (Ra→Rb) and "backward" (RbH→RaH) shuttling motions (vertical processes in Figure 2) of the molecular ring, which occur, respectively, upon deprotonation and reprotonation of the ammonium/amine recognition site on the axle (horizontal processes in Figure 2), were found to be $0.72s^{-1}$ and $40s^{-1}$ at $293°K$, respectively.

4.1 Modeling the Rotaxane RaH in nanoκ Calculus

The nanoκ Calculus molecules. Figure 3 illustrates the nanoκ Calculus modeling of the rotaxane RaH. We use four species:

- *Nh* models the ammonium/amine station of the rotaxane: it has one field h and two sites *ring* and *axle*;
- *Axle* models the spacer between the two stations: it has two fields s and h and three sites *nh*, *bipy*, and *ring*;
- *Bipy* models the bipyridinium station: it has one field h and two sites *ring* and *axle*;
- *Ring* models the crown ether ring: it has no field and one site *link*.

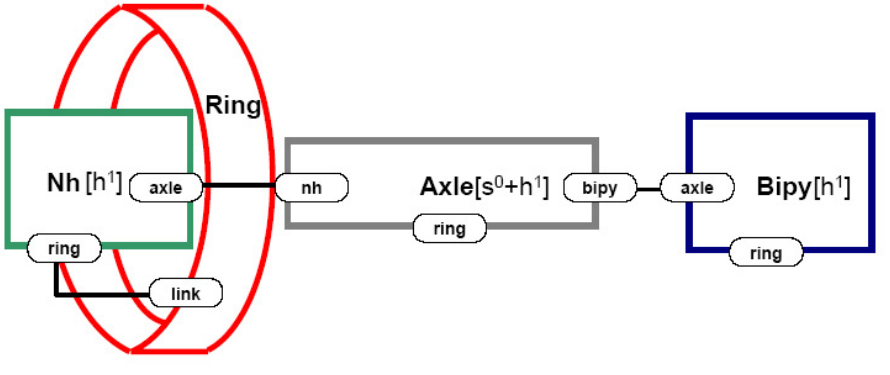


Fig. 3. Initial state of the Rotaxane RaH in nanoκ Calculus

The pairs of sites *axle* of *Nh* and *nh* of *Axle*, and *axle* of *Bipy* and *bipy* of *Axle* are always linked in our modeling. They model the covalent bonds maintaining the structural integrity of the axle. Exactly one site *ring* of *Nh*, *Bipy*, and *Axle* is linked at a given moment at *link* of *Ring*. The first two cases respectively model the "stable" RaH and Rb states of Figure 2 in which the ring is steadily located around the *Nh* or the *Bipy* molecules, respectively. The last case models the "unstable" states; these are the Ra and RbH states of Figure 2 in which the ring is not steadily located. In order to distinguish between the Ra and RbH states, we use the field s of the *Axle*: it holds the value 0 if the ring is around the *Nh* (Ra state), 1 if it is around the *Bipy* (RbH state).

Ammonium and amine functions have different chemical nature but can be seen as protonated and deprotonated version of the same species. Thus we model both by the same nanoκ Calculus species *Nh*. Its field h is used to record the presence or absence of a proton on *Nh*: its value is 1 if it is protonated, and 0 otherwise.

As *Ring*'s movements are triggered by protonations and deprotonations due to acid-base reactions, we also need to have acid and base molecules in our modeling. We consider the species *Acid* and *Base* both with one field h having value 1 in case the acid/base molecule holds the proton to be exchanged, 0 otherwise (for instance $Acid[h^1]$ and $Base[h^0]$ are respectively an acid molecule ready to give a proton and a base molecule ready to receive a proton).

The initial state for rotaxane RaH is thus modeled by the term:

$$Nh[h^1](axle^s + ring^x) \ , \quad Axle[s^0 + h^1](nh^s + bipy^r + ring) \ ,$$
$$Bipy[h^1](axle^r + ring) \ , \quad Ring(link^x)$$

graphically depicted in Figure 3.

Note that the *Nh* is initially protonated (and this information is present also in the *Axle* and the *Bipy*), the *Axle* is bound to the *Nh* and the *Bipy*, and the *Ring* is bound to the *Nh*.

The nanoκ Calculus reactions. We now present the reactions used in our modeling. Reactions 1, 2, 7 and 8 are presented with a double arrow (that are *reversible* reactions). Formally they correspond to two nanoκ Calculus reactions, one achieved reading the reaction from left to right considering the rate over the arrow, and another one achieved reading it from right to left considering the rate below. In this section we do not consider numerical values of rates, this is detailed in part 4.2.

A base can get the proton of a protonated *Nh*, and a *Nh* can get a proton from an acid. These acid-base reactions are reversible. Reactions 1 and 2 model this phenomena. The systems corresponding to the left-hand side and right-hand side coexist, even if one can be much predominant according to the ratio $nh\_base/base\_nh$ (and $acid\_nh/nh\_acid$).

$$Nh[h^1], Base[h^0] \underset{base\_nh}{\overset{nh\_base}{\rightleftharpoons}} Nh[h^0], Base[h^1] \tag{1}$$

$$Nh[h^0], Acid[h^1] \underset{nh\_acid}{\overset{acid\_nh}{\rightleftharpoons}} Nh[h^1], Acid[h^0] \tag{2}$$

The protonation state of the molecule *Nh* needs to be known by *Bipy* because it affects its interaction with *Ring*. Reactions 3 and 4 achieve this by passing information from *Nh* to *Bipy* through *Axle*. These updates are instantaneous because the reactions have infinite rates (this is relevant for the correctness of our simulation, since these reactions have no counterpart in chemistry).

if $(\alpha \neq \beta)$

$$Nh[h^\alpha](axle^s), Axle[h^\beta](nh^s) \overset{\infty}{\rightarrow} Nh[h^\alpha](axle^s), Axle[h^\alpha](nh^s) \tag{3}$$

and:

$$Axle[h^\alpha](bipy^r), Bipy[h^\beta](axle^r) \overset{\infty}{\rightarrow} Axle[h^\alpha](bipy^r), Bipy[h^\alpha](axle^r) \tag{4}$$

We achieve the modeling of *Ring* movements in two steps. Firstly the instantaneous reactions to deprotonation/reprotonation (reactions 5–8), and secondly the actual *Ring* shuttling (reactions 9 and 10). As reactions 5–8 represent immediate consequences of deprotonation or reprotonation of *Nh*, they have infinite rates. Reactions 9 and 10 are reversible, because the *Ring* is susceptible to return to the previous station due to the Brownian motion.

$$Nh[h^0](axle^s + ring^x), Axle[s^0](nh^s + ring) \overset{\infty}{\rightarrow} Nh[h^0](axle^s + ring), Axle[s^0](nh^s + ring^x) \tag{5}$$

$$Bipy[h^1](axle^r + ring^x), Axle[s^1](biax^r + ring) \overset{\infty}{\rightarrow} Bipy[h^1](axle^r + ring), Axle[s^1](biax^r + ring^x) \tag{6}$$

$$Axle[s^0](nh^s + ring^x), Nh[h^1](axle^s + ring) \overset{\infty}{\rightarrow} Axle[s^0](nh^s + ring), Nh[h^1](axle^s + ring^x) \tag{7}$$

$$Axle[s^1](nh^s + ring^x), Bipy[h^0](axle^s + ring) \overset{\infty}{\rightarrow} Axle[s^1](nh^s + ring), Bipy[h^0](axle^s + ring^x) \tag{8}$$

$$Axle[s^0](bipy^r + ring^x), Bipy[h^0](axle^r + ring) \underset{unlink\_bipy}{\overset{link\_bipy}{\rightleftharpoons}} Axle[s^1](bipy^r + ring), Bipy[h^0](axle^r + ring^x) \tag{9}$$

$$Axle[s^1](nh^s + ring^x), Nh[h^1](axle^s + ring) \underset{unlink\_nh}{\overset{link\_nh}{\rightleftharpoons}} Axle[s^0](nh^s + ring), Nh[h^1](axle^s + ring^x) \tag{10}$$

4.2 Simulation Results

It is not difficult to verify that the above modeling of rotaxane RaH in `nano`κ Calculus yields a stictly markovian IMC system. Therefore we may safely downgrade it to a CTMC system that we use to simulate *in silico* the behaviour of the rotaxane RaH.

As previously discussed the rates for the ring movements are respectively $link\_bipy = 0.72$ and $link\_nh = 40$. On the basis of the estimated equilibrium constants, the rates for the reverse reactions are quantified two orders of magnitude smaller, i.e. $unlink\_bipy = 0.0072$ and $unlink\_nh = 0.4$.

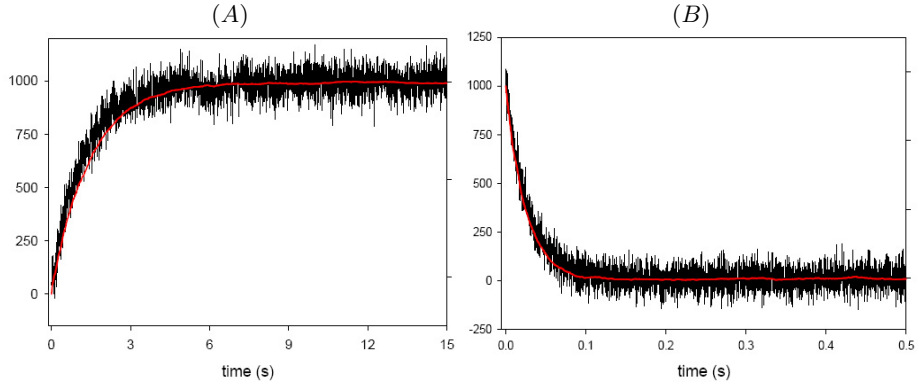


Fig. 4. Comparing the simulations *in silico* with the experiments *in vitro*. Grey traces: number of *Ring*s located around *Bipy*s during the "forward" Ra→Rb (part A) and the "backward" RbH→RaH (part B). Black traces: UV absorbance changes observed upon the occurrence of the same respective shuttling processes.

The aim of the first two simulations depicted in Figure 4 is to check whether the experimentation *in silico* can reproduce the results observed in *in vitro* [8]. The techniques used for the *in vitro* experimentation did not permit to observe and quantify the deprotonation/reprotonation rates (this is not surprising as these are very fast acid-base reactions). Thus, in the simulation we have considered instantaneous deprotonation/reprotonation, i.e. $nh\_base = acid\_nh = \infty$ and $base\_nh = nh\_acid = 0$. In both simulations, we have considered 1000 rotaxanes: in the first one we have simulated deprotonation and "forward" (Ra→Rb) shuttling, in the second one reprotonation and "backward" (RbH→RaH) shuttling. In the first simulation the shuttling phase is completed in around 6 seconds, while in the second one in 0.1 seconds; this is a consequence of the different rates of the two directions of shuttling. Very remarkably, simulated data are in strike agreement with the experimental results.

After these initial encouraging results, we have decided to use the *in silico* simulation techniques to provide a comprehensive view of the overall reactions depicted in Figure 2, simulating also the deprotonation/reprotonation phases not observed in the *in silico* experimentation. More precisely, the aim of this second group of simulations was to either validate or invalidate the assumption according to which deprotonation/reprotonation can be considered "instantaneous" with respect to the shuttling time. To this aim, we have simulated deprotonation/reprotonation under two different concentrations of rotaxanes. In fact, this is a bimolecular reaction whose rate is influenced by the concentration of

the reactants. For instance, at the concentration considered in [8], i.e. $10^{-5}M$, assuming 1000 instances of rotaxane and base/acid, a plausible rate for deprotonation/reprotonation is $2 \times 10^{2}s^{-1}$ (with reverse reaction rate on the order of $2 \times 10^{-5}s^{-1}$) while at the concentration $10^{-8}M$ it is $0.2s^{-1}$ (with reverse reaction on the order of $0.2 \times 10^{-7}s^{-1}$).

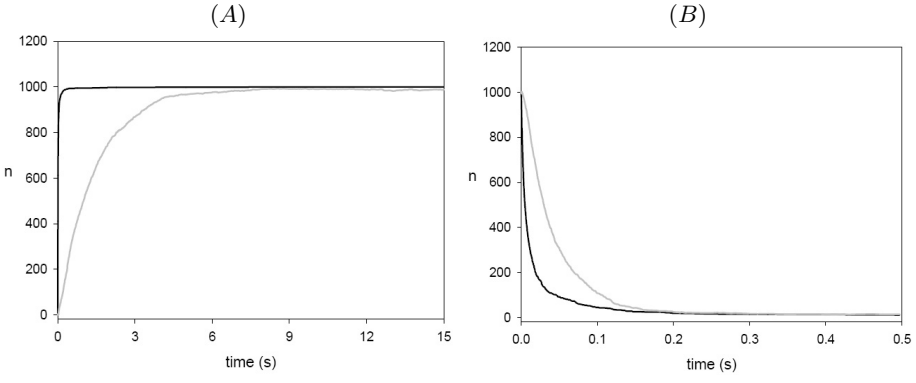


Fig. 5. Number of *Ring*s located around *Bipy*s (grey trace) and number of protonated rotaxanes (black trace) during the "forward" shuttling in the presence of base molecules (part A) and the "backward" shuttling in the presence of acid molecules (part B)

We have performed the two simulations, namely deprotonation with subsequent "forward" shuttling and reprotonation with subsequent "backward" shuttling, considering the two different concentrations.

The results at concentration $10^{-5}M$ are not reported in the paper as they essentially confirm the validity of the "instantaneous" deprotonation/reprotonation assumption. We report in Figure 5 the results for concentration $10^{-8}M$ as they are definitely more interesting; the rings start moving before the deprotonation/reprotonation phase is over. This proves that in the rotaxane RaH the stimulus and the subsequent shuttling could interplay, and this opens interesting scenarios that requires further investigation. For instance, it could be the case that using weak acid/base molecules (for which the ratio between the deprotonation/reprotonation rate and the reverse rate is smaller) the interplay between the stimulus and the shuttling could give rise to currently unknown emerging behaviours.

References

1. Ashton, P.R., Ballardini, R., Balzani, V., Credi, A., Baxter, I., Fyfe, M.C.T., Gandolfi, M.T., Gómez-López, M., Martínez-Díaz, M.-V., Piersanti, A., Spencer, N., Stoddart, J.F., Venturi, M., White, A.J.P., Williams, D.J.: Acid-base controllable molecular shuttles. Journal of Am. Chem. Soc. 120, 11932–11942 (1998)
2. Badjic, J.D., Balzani, V., Credi, A., Silvi, S., Stoddart, J.F.: A molecular elevator. Science 303, 1845–1849 (2004)

3. Balzani, V., Credi, A., Venturi, M.: Molecular devices and machines – concepts and perspectives for the nano world. Wiley-VCH, Weinheim (2007)
4. Cardelli, L.: On process rate semantics (2007)
5. Cardelli, L., Phillips, A.: Spim homepage At (2006), research.microsoft.com/aphillip/spim/
6. Champin, B., Mobian, P., Sauvage, J.-P.: Transition metal complexes as molecular machine prototypes. Chemical Society Reviews 36(2), 358–366 (2006)
7. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science 325(1), 69–110 (2004)
8. Garaudée, S., Silvi, S., Venturi, M., Credi, A., Flood, A.H., Stoddart, J.F.: Shuttling dynamics in an acid-base-switchable [2]rotaxane. Chem. Phys. Chem. 6, 2145 (2005)
9. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81, 2340–2361 (1977)
10. Gilmore, S., Hillston, J.: The pepa workbench: a tool to support a process algebra-based approach to performance modelling. In: Proc. of the 7th int. conference on Computer performance evaluation: modelling techniques and tools, Secaucus, NJ, USA, pp. 353–368. Springer, New York (1994)
11. Hermanns, H.: Interactive Markov Chains: The quest for quantitative quality. LNCS, vol. 2428. Springer, Heidelberg (2002)
12. Huang, T.-J., Brough, B., Ho, C.-M., Liu, Y., Flood, A.H., Bonvallet, P.A., Tseng, H.-R., Stoddart, J.F., Baller, M., Magonov, S.: A nanomechanical device based on linear molecular motors. Applied Physics Letters 85(22), 5391–5393 (2004)
13. Jimenez, M.C., Dietrich-Buchecker, C., Sauvage, J.-P.: Towards synthetic molecular muscles: Contraction and stretching of a linear rotaxane dimer. Angew. Chem. Int. Ed. 39(18), 3284–3287 (2000)
14. Laneve, C., Tarissan, F.: A simple calculus of protein and cells. In: Proceeding of MeCBIC'06. ENTCS, vol. 1677 (2007)
15. Martínez-Díaz, M.-V., Spencer, N., Stoddart, J.F.: The self-assembly of a switchable [2]rotaxane. Angew. Chem. Int. Ed. Engl. 36, 1904–1907 (1997)
16. Milner, R.: Communicating and mobile systems: the π-calculus. Cambridge University Press, Cambridge (1999)
17. Plotkin, G.D.: A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus (1981)
18. Priami, C.: Stochastic pi-calculus. Computer Journal 38(7), 578–589 (1995)
19. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters 80, 25–31 (2001)
20. Sauvage, J.-P., Dietrich-Bucheker, C.O. (eds.): Molecular catenanes, rotaxanes and knots. Wiley-VCH, Weinheim (1999)

Efficient, Correct Simulation of Biological Processes in the Stochastic Pi-calculus

Andrew Phillips and Luca Cardelli

Microsoft Research, 7 JJ Thomson Avenue, CB3 0FB Cambridge UK
{andrew.phillips,luca}@microsoft.com

Abstract. This paper presents a simulation algorithm for the stochastic π-calculus, designed for the efficient simulation of biological systems with large numbers of molecules. The cost of a simulation depends on the number of species, rather than the number of molecules, resulting in a significant gain in efficiency. The algorithm is proved correct with respect to the calculus, and then used as a basis for implementing the latest version of the SPiM stochastic simulator. The algorithm is also suitable for generating graphical animations of simulations, in order to visualise system dynamics.

1 Introduction

In recent years, there has been considerable research on designing programming languages for complex parallel computer systems. Interestingly, some of this research is also applicable to biological systems, which are typically highly complex and massively parallel. In particular, a mathematical programming language known as the stochastic π-calculus has recently been used to model and simulate a range of biological systems [7,12,13]. The calculus allows the components of a biological system to be modelled independently, rather than modelling the individual reactions. This allows large models to be constructed by composition of simple components [2]. The calculus also facilitates mathematical analysis of systems using a range of established techniques, which could eventually shed light on some of the fundamental properties of biological systems. Various stochastic simulators have been developed for the calculus [13,9,1], in order to perform virtual experiments on biological system models. Such *in silico* experiments can be used to formulate testable hypotheses on the behaviour of biological systems, as a guide to future experimentation *in vivo*.

Currently available simulators for the stochastic π-calculus are implemented based on standard theory of chemical kinetics, using an adaptation of the Gillespie algorithm [5]. This algorithm has the distinct advantage of being mathematically exact, enabling accurate simulation of biological models. Unfortunately, the algorithm is also highly computationally intensive, particularly when simulating large models. As a result, there has been considerable research on optimisations for the Gillespie algorithm, resulting in a plethora of alternatives, both exact and approximate [4,6,15]. Like the original algorithm, these alternatives are defined

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 184–199, 2007.

in terms of systems of chemical reactions, the *de facto* standard for biological modelling. This reaction view of systems differs in many ways from the component view of the stochastic π-calculus. As a result, techniques for efficient simulation of chemical reactions cannot be directly applied to the stochastic π-calculus, but need to be adapted to account for the differences between the two formalisms [10]. Given these differences, and given the importance of efficiency in the stochastic simulation of biological models, research on efficient simulation algorithms for the stochastic π-calculus seems of interest. There has already been substantial research on efficient implementation techniques for variants of the π-calculus, in the context of programming languages for parallel computer systems [16]. However, this research does not take into account the specific properties of biological systems, which differ from most computer systems in fundamental ways. One key difference is that biological systems are often composed of large numbers of processes with identical behaviour, such as thousands of proteins of the same type.

This paper presents a simulation algorithm for the stochastic π-calculus, designed for the efficient simulation of biological systems with large numbers of molecules. The paper is structured as follows. Section 2 illustrates the principle of the simulation algorithm with the help of a biological example. Section 3 presents the full definition of the algorithm, and Sec. 4 outlines a proof of correctness with respect to the stochastic π-calculus. Finally, Sec. 5 shows how the algorithm can be mapped to executable program code, in order to implement a stochastic simulator.

2 Biological Example

This section introduces the simulation algorithm for the stochastic π-calculus, with the help of a biological example. The example describes a system of three genes with negative control that mutually repress each other, as presented in [11]. The system consists of an environment, which contains definitions for a $Gene(a, b)$ and a $Protein(b)$, together with a top-level process, which contains three genes executing in parallel :

$$\begin{aligned}&\{Gene(a,b) = \tau_t.(Gene(a,b) \mid Protein(b)) + ?a.\tau_u.Gene(a,b),\\ &Protein(b) = !b.Protein(b) + \tau_d\}\\ &\vdash\\ &Gene(a,b) \mid Gene(b,c) \mid Gene(c,a)\end{aligned}$$

A $Gene(a, b)$ is parameterised by its promoter region a, together with the promoter region b that is recognised by its transcribed proteins. The gene can perform one of two actions, represented as a choice $(+)$. Either it can transcribe a $Protein(b)$ by doing a stochastic delay τ_t, after which the new protein is executed in parallel with the gene, or it can block by doing an input $?a$ on its promoter region a and then unblock by doing a stochastic delay τ_u. A $Protein(b)$ can repeatedly do an output $!b$ on the promoter region b, or it can degrade by doing a stochastic delay τ_d. According to the reduction rules of the calculus, the input

$?b$ of a $Gene(b,c)$ can interact with the output $!b$ of a corresponding protein, becoming blocked as a result. The three genes in the system can mutually repress each other, since $Gene(a,b)$ produces proteins that can block $Gene(b,c)$, which produces proteins that can block $Gene(c,a)$, which produces proteins that can block $Gene(a,b)$, completing the cycle. Stochastic behaviour is incorporated into the system by associating each of the channels a, b, c with corresponding interaction rates given by $\rho(a), \rho(b), \rho(c)$, respectively, and by associating each of the delays τ_t, τ_u, τ_d with corresponding delay rates given by t, u, d. These rates are used to calculate the probabilities of all the reactions in the system, where the probability of a reaction is proportional to its rate.

The above system is simulated by encoding it to a system $E \vdash V$ of the stochastic π-machine, which consists of a machine environment E and a machine term V:

$$
\begin{array}{l}
\{Gene(a,b) = \tau_t.(Gene(a,b) \mid Protein(b)) + ?a.X(a,b), \\
X(a,b) = \tau_u.Gene(a,b), \\
Protein(b) = !b.Protein(b) + \tau_d\} \\
\vdash \\
\emptyset, \{a \mapsto (1{,}0{,}0{,}0), b \mapsto (1{,}0{,}0{,}0), c \mapsto (1{,}0{,}0{,}0), t \mapsto (3{,}3t)\}, \\
\{Gene(a,b) \mapsto 1, \{t \mapsto 1, a \mapsto (1{,}0)\}, \tau_t.(Gene(a,b) \mid Protein(b)) + ?a.X(a,b), \\
Gene(b,c) \mapsto 1, \{t \mapsto 1, b \mapsto (1{,}0)\}, \tau_t.(Gene(b,c) \mid Protein(c)) + ?b.X(b,c), \\
Gene(c,a) \mapsto 1, \{t \mapsto 1, c \mapsto (1{,}0)\}, \tau_t.(Gene(c,a) \mid Protein(a)) + ?c.X(c,a)\}
\end{array}
$$

The machine environment E is similar to a calculus environment, with the additional constraint that each choice of one or more actions must be associated with a corresponding identifier. In order to satisfy this constraint, the encoding creates a new definition $X(a,b)$, which keeps track of the number of genes in a blocked state. Note that the constraint is not enforced at the calculus level, since this would be too much of syntactic burden. Instead, the extra definitions are created by the encoding.

The machine term V consists of a set of channels Z, a store of reactions S and a heap of species H. The set Z denotes the set of all the private channels in the system, which is empty in this example. The store S records the apparent rate of all the delays and channels in the system. The apparent rate of a delay of rate r is given by the number of possible delays τ_r multiplied by r. The apparent rate of a channel x is given by the number of possible interactions on the channel multiplied by $\rho(x)$. This information is recorded in the Store S, where each delay is mapped to the number of delays and the apparent rate of the delay, and each channel is mapped to the number of inputs, outputs, mixed interactions (i.e. the number of pairs of inputs and outputs that cannot interact with each other) and the apparent rate of the channel. In the above example, there are initially three delays of rate t and one input on each channel a, b, c. The heap H records information about each species that is currently being simulated, including the population of the species, the choice of actions that the species can perform and the number of each type of action. Initially there are three species in the system, $Gene(a,b)$, $Gene(b,c)$ and $Gene(c,a)$, where each gene with a given set

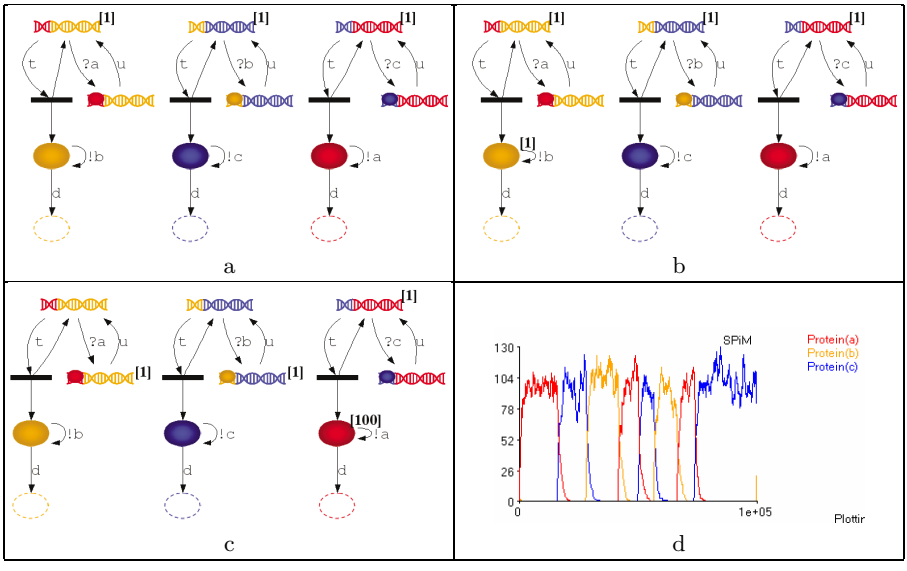


Fig. 1. Graphical representation of a network of three genes with inhibitory control that mutually repress each other. It is assumed that $\rho(a) = \rho(b) = \rho(c)$ and $\rho(a) \gg t \gg d \gg u$. Initially there is one copy of each gene (a), and one of the genes then transcribes a protein (b). After a sequence of reduction steps, two of the genes become blocked, and the third gene produces 100 proteins (c). The mutual repression of genes gives rise to alternate oscillations of protein levels, as shown in the simulation plot (d), where the vertical axis represents the number of proteins and the horizontal axis represents the simulation time. The results were obtained with $\rho(a) = 1.0$, $t = 0.1$ $d = 0.001$ and $u = 0.0001$.

of parameters denotes a separate species. The $Gene(a, b)$ can do a delay at rate t or an input on channel a, and similarly for the remaining genes in the system.

A corresponding graphical representation for this system is shown in Fig. 1(a) based on [11], where a separate graph is drawn for each gene. Each shape in the graph represents a species, and each labelled edge represents an action that the species can perform. Multiple edges from a species correspond to a choice, while multiple edges from a horizontal bar correspond to a parallel composition.

In order to execute this system, the stochastic π-machine chooses one of the possible reactions using an adaptation of the Gillespie algorithm [5], where the probability of a reaction is proportional to its rate. Initially, the machine can do one of three delay reactions with rate t, where the apparent rate of the delay is $3t$. The machine chooses one of these delays with equal probability. Suppose the $Gene(a, b)$ is chosen to perform the delay. An additional $Protein(b)$ is produced, giving rise to the following machine term, which corresponds to Fig. 1(b):

$$\emptyset, \{a \mapsto (1{,}0{,}0{,}0), b \mapsto (1{,}1{,}0{,}\rho(b)), c \mapsto (1{,}0{,}0{,}0), d \mapsto (1{,}d), t \mapsto (3{,}3t)\},$$
$$\{Gene(a,b) \mapsto 1, \{t \mapsto 1, a \mapsto (1{,}0)\}, \tau_t.(Gene(a,b) \mid Protein(b)) + ?a.X(a,b),$$
$$Protein(b) \mapsto 1, \{d \mapsto 1, b \mapsto (0{,}1)\}, !b.Protein(b) + \tau_d,$$
$$Gene(b,c) \mapsto 1, \{t \mapsto 1, b \mapsto (1{,}0)\}, \tau_t.(Gene(b,c) \mid Protein(c)) + ?b.X(b,c),$$
$$Gene(c,a) \mapsto 1, \{t \mapsto 1, c \mapsto (1{,}0)\}, \tau_t.(Gene(c,a) \mid Protein(a)) + ?c.X(c,a)\}$$

The Gillespie algorithm is then used to execute the next reaction. Assuming $\rho(b) \gg t$ there is a high likelihood that a reaction on b will be chosen, which blocks $Gene(b,c)$. Subsequently, a $Protein(a)$ is transcribed, which blocks $Gene(a,b)$. Since both $Gene(b,c)$ and $Gene(a,b)$ are blocked, no more $Protein(c)$ or $Protein(b)$ are produced. Eventually 100 copies of $Protein(a)$ are produced, giving rise to the following machine term, which corresponds to Fig. 1(c):

$$\emptyset, \{a \mapsto (0{,}100{,}0{,}0), b \mapsto (1{,}0{,}0{,}0), c \mapsto (1{,}0{,}0{,}0), t \mapsto (1{,}t), u \mapsto (2{,}2u), d \mapsto (100{,}100d)\},$$
$$\{X(a,b) \mapsto 1, \{u \mapsto 1\}, \tau_u.Gene(a,b),$$
$$X(b,c) \mapsto 1, \{u \mapsto 1\}, \tau_u.Gene(b,c),$$
$$Gene(c,a) \mapsto 1, \{t \mapsto 1, c \mapsto (1{,}0)\}, \tau_t.(Gene(c,a) \mid Protein(a)) + ?c.\tau_u.X(c,a),$$
$$Protein(a) \mapsto 100, \{d \mapsto 1, a \mapsto (0{,}1)\}, !a.Protein(a) + \tau_d\}$$

This represents the first oscillation cycle from the simulation results of Fig. 1(d). Note how the graphical representation relies on the ability to count the number of copies of each species, in order to label the corresponding node with its population. The graphs are generated using a translation to DOT syntax [3] based on [11]. The resulting sequence of pictures can also be used to produce a 3D animation of the simulation, as shown in [9]. Note that the simulation algorithm does not require the countable species of a simulation to be known beforehand. Rather, the populations of species are grouped on-the-fly according to the species name and parameters. In the above example, at the start of the simulation there are three definitions for $Gene(a,b)$, $X(a,b)$ and $Protein(a)$ in the environment, and three species $Gene(a,b)$, $Gene(b,c)$ and $Gene(c,a)$ in the heap. As the simulation proceeds, additional species are dynamically created in the heap by instantiating the definitions with different parameters. In the general case, a new species is dynamically created for each new combination of parameters, which is potentially unbounded.

3 Simulation Algorithm

This section presents a formal definition of the stochastic π-machine (SPiM). The syntax of processes and environments in SPiM is given in Definition 1. This is a subset of the syntax of the stochastic π-calculus (Definition 10), with the additional constraint that each choice of one or more actions can only occur at the top level of a definition, as in [11]. Stochastic behaviour is incorporated into the system by associating each channel x with a corresponding interaction rate given by $\rho(x)$, and by associating each delay τ_r with a corresponding delay rate r. Each rate characterises an exponential distribution, such that the probability

$$
\begin{array}{rcll}
P, Q ::= & \mathbf{0} & \text{Null} \\
| & X(\tilde{n}) & \text{Instance} \\
| & P \mid Q & \text{Parallel} \\
| & \nu x\, P & \text{Restriction} \\
\\
M ::= & \mathbf{0} & \text{Null} \\
| & \pi.P + M & \text{Action} \\
C ::= & \nu \tilde{n}\, M & \text{Choice}
\end{array}
\qquad
\begin{array}{rcll}
E ::= & \emptyset & \text{Empty} \\
| & E, X(\tilde{m}) = P & \text{Process} \\
| & E, X(\tilde{m}) = C & \text{Choice} \\
\\
\pi ::= & ?x(\tilde{m}) & \text{Input} \\
| & !x(\tilde{n}) & \text{Output} \\
| & \tau_r & \text{Delay}
\end{array}
$$

Definition 1. *Syntax of processes and environments in* SPiM. For convenience, D is used to denote the body of a definition, which can be a process P or a choice C. For each definition of the form $X(\tilde{m}) = D$ it is assumed that $\mathrm{fn}(D) \subseteq \tilde{m}$.

$$
\begin{array}{rcll}
S ::= & \emptyset & \text{Empty} \\
| & S, r \mapsto (\mathrm{Delay}_r, a_r) & \text{Delay} \\
| & S, x \mapsto (\mathrm{In}_x, \mathrm{Out}_x, \mathrm{Mix}_x, a_x) & \text{Channel}
\end{array}
\qquad
\begin{array}{rcll}
U ::= & \emptyset & \text{Empty} \\
| & U, r \mapsto \mathrm{Delay}_r & \text{Delay} \\
| & U, x \mapsto (\mathrm{In}_x, \mathrm{Out}_x) & \text{Channel}
\end{array}
$$

Definition 2. *Syntax of stores and substores in* SPiM..

$$
\begin{array}{rcll}
V ::= & Z, S, H & \text{Term} \\
I ::= & X(\tilde{n}) & \text{Instance}
\end{array}
\qquad
\begin{array}{rcll}
H ::= & \emptyset & \text{Empty} \\
| & H, I \mapsto (i, U, C) & \text{Species}
\end{array}
$$

Definition 3. *Syntax of terms in* SPiM. For each mapping $I \mapsto (i, U, C)$ it is assumed that $I \equiv C$ according to Definition 12(19) and $U = \mathrm{Sub}(C)$ according to Definition 4.

$$
\begin{aligned}
\mathrm{In}_x(?x(\tilde{m}).P + M) &\triangleq 1 + \mathrm{In}_x(M) \\
\mathrm{In}_x(\pi.P + M) &\triangleq \mathrm{In}_x(M) \textbf{ if } \pi \neq ?x(\tilde{m}) \\
\mathrm{Out}_x(!x(\tilde{n}).P + M) &\triangleq 1 + \mathrm{Out}_x(M) \\
\mathrm{Out}_x(\pi.P + M) &\triangleq \mathrm{Out}_x(M) \textbf{ if } \pi \neq !x(\tilde{n}) \\
\mathrm{Delay}_r(\tau_r.P + M) &\triangleq 1 + \mathrm{Delay}_r(M) \\
\mathrm{Delay}_r(\pi.P + M) &\triangleq \mathrm{Delay}_r(M) \textbf{ if } \pi \neq \tau_r
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{Sub}(\nu \tilde{n}\, M) \triangleq\ & \{x \mapsto (i, o) \mid i = \mathrm{In}_x(M) \wedge o = \mathrm{Out}_x(M) \wedge (i, o) \neq (0, 0) \wedge x \notin \tilde{n}\} \\
& \cup \{r \mapsto d \mid d = \mathrm{Delay}_r(M) \wedge d \neq 0\}
\end{aligned}
$$

Definition 4. *Creating a substore in* SPiM, *where* $\mathrm{In}_x(\mathbf{0}) = \mathrm{Out}_x(\mathbf{0}) = \mathrm{Delay}_r(\mathbf{0}) = 0$.

of a reaction with rate r occurring within time t is given by $F(t) = 1 - e^{-rt}$. The average duration of the reaction is given by the mean $1/r$ of this distribution.

The syntax of machine terms is given in Definitions 2 and 3. A machine term V consists of a set of private channels Z, a store S and a heap H. The store S records the activity and apparent rate of all the unguarded delays and channels in the system. The activity of a delay with rate r is given by Delay_r, which records the total number of delays of rate r. The apparent rate a_r of the delay is equal to $r \times \text{Delay}_r$. The activity of a channel x is given by the triple $\text{In}_x, \text{Out}_x, \text{Mix}_x$, which records the total number of inputs, outputs and mixed interactions on x, respectively, where the number of mixed interactions denotes the number of pairs of inputs and outputs that cannot interact on x. The apparent rate a_x of the channel is equal to $\rho(x) \times (\text{In}_x \times \text{Out}_x - \text{Mix}_x)$. The heap H keeps track of the number of copies of identical species in the system, and consists of zero or more mappings from species I to triples (i, U, C), where the number i records the population of the species, the choice C records the actions that the species can perform, and the substore U records the number of inputs and outputs on each channel in C, together with the number of each type of delay in C. The notation $H(I)$ denotes the values associated to I in the heap H, as usual. A substore U is created from a choice C according to Definition 4.

The system is executed according to the reduction rules of the stochastic π-machine, described in Definition 9. The rules rely on a number of auxiliary functions, given in Definitions 5–8. The expression $S \oplus U$ adds a substore U to a store S, as described in Definition 5. This is used to update the store each time the population of a species changes during a simulation. The number of delays, inputs and outputs in the species are added to the totals in the store, where the number of mixed interactions is calculated incrementally. Subtraction $S \ominus U$ is defined in a similar way.

The expression $(Z, S, H) \oplus \{I \mapsto C\}$ adds a species I with body C to a term (Z, S, H), as described in Definition 6. If a binding (i, U, C) for I is already present in the heap then the population i of the species is incremented (1). Otherwise, a new binding (i, U, C) for I is created, where the substore U denotes the total activity of the species, given by $\text{Sub}(C)$, and the population of the species is set to 1 (2). Note that whenever a new species is added to a term, the substore U of the species needs to be added to the store S. The expression $(Z, S, H) \ominus \{I \mapsto C\}$ removes a species I with body C from a term (Z, S, H) (3).

The expression $V \oplus P$ adds a machine process P to a machine term V, as described in Definition 7. The null process $\mathbf{0}$ is discarded (4). If an instance $X(\tilde{n})$ is defined as a choice $X(\tilde{m}) = C$ then the term is updated with a mapping from $X(\tilde{n})$ to the body of the definition, in which the parameters $\tilde{m}$ are instantiated with the values $\tilde{n}$ (5). If an instance $X(\tilde{n})$ is defined as a process $X(\tilde{m}) = P$ then the body of the definition is added to the term, in which the parameters $\tilde{m}$ are instantiated with the values $\tilde{n}$ (6). A parallel composition $P \mid Q$ is split so that each process is added separately (7). A restriction $\nu x\, P$ is added to a term by replacing x with a fresh channel y and adding this to the set of private channels Z (8).

$$
\begin{aligned}
S \oplus \emptyset &\triangleq S \\
S \oplus (U, r \mapsto d) &\triangleq (S \oplus U), r \mapsto (d, (d \times r)) \text{ if } S(r) = \emptyset \\
(S, r \mapsto (d, a)) \oplus (U, r \mapsto d') &\triangleq (S \oplus U), r \mapsto (d + d', a + d' \times r) \\
S \oplus (U, x \mapsto (i, o)) &\triangleq (S \oplus U), x \mapsto (i, o, i \times o, 0) \text{ if } S(x) = \emptyset \\
(S, x \mapsto (i, o, m, a)) \oplus (U, x \mapsto (i', o')) &\triangleq (S \oplus U), x \mapsto (i + i', o + o', m + i' \times o', a') \\
&\textbf{if } a' = a + (i \times o' + i' \times o) \times \rho(x)
\end{aligned}
$$

Definition 5. *Adding a substore to a store in* SPiM.

$$
\begin{aligned}
(Z, S, H) \oplus \{I \mapsto C\} &\triangleq Z, (S \oplus U), H\{I \mapsto (i{+}1, U, C)\} \textbf{ if } H(I) = (i, U, C) && (1)\\
(Z, S, H) \oplus \{I \mapsto C\} &\triangleq Z, (S \oplus U), H\{I \mapsto (1, U, C)\} \textbf{ if } H(I) = \emptyset, U = \mathrm{Sub}(C) && (2)\\
(Z, S, H) \ominus \{I \mapsto C\} &\triangleq Z, (S \ominus U), H\{I \mapsto (i{-}1, U, C)\} \textbf{ if } H(I) = (i, U, C), i > 0 && (3)
\end{aligned}
$$

Definition 6. *Adding and removing a species from a term in* SPiM.

$$
\begin{aligned}
V \oplus \mathbf{0} &\triangleq V && (4)\\
V \oplus X(\tilde{n}) &\triangleq V \oplus \{X(\tilde{n}) \mapsto C_{\{\tilde{n}/\tilde{m}\}}\} \textbf{ if } X(\tilde{m}) = C && (5)\\
V \oplus X(\tilde{n}) &\triangleq V \oplus P_{\{\tilde{n}/\tilde{m}\}} \textbf{ if } X(\tilde{m}) = P && (6)\\
V \oplus (P \mid Q) &\triangleq V \oplus P \oplus Q && (7)\\
(Z, S, H) \oplus (\nu x\, P) &\triangleq (Z \cup \{y\}, S, H) \oplus P_{\{y/x\}} \textbf{ if } y \text{ fresh} && (8)
\end{aligned}
$$

Definition 7. *Adding a process to a term in* SPiM.

1. Calculate $a_0 = \sum_{i=1}^{N} a_i$ for all the reactions $\theta_1, \ldots, \theta_N$ in the domain of S
2. Generate two random numbers $n_1, n_2 \in [0, 1]$ and calculate t, μ such that:

$$t = (1/a_0) \ln(1/n_1)$$

$$\sum_{i=1}^{\mu-1} a_i < n_2 a_0 \leq \sum_{i=1}^{\mu} a_i$$

3. $Gillespie(Z, S, H) = \theta_\mu, t$

Definition 8. *Choosing the next reaction in* SPiM *using the Gillespie algorithm* [5].

$$\frac{r, t = Gillespie(V) \qquad V' = V \ominus \{I \mapsto \nu\tilde{m}\,(\tau_r.P + M)\}}{V \xrightarrow{r} V' \oplus (\nu\tilde{m}\,P)} \tag{9}$$

$$\frac{x, t = Gillespie(V) \qquad V' = V \ominus \{I_1 \mapsto \nu\tilde{m}_1\,(!x(\tilde{n}).P_1 + M_1)\} \ominus \{I_2 \mapsto \nu\tilde{m}_2\,(?x(\tilde{m}).P_2 + M_2)\}}{V \xrightarrow{\rho(x)} V' \oplus \nu\tilde{m}_1\,\nu\tilde{m}_2\,(P_1 \mid P_{2\,\{\tilde{n}/\tilde{m}\}})} \tag{10}$$

$x \notin \tilde{m}_1 \cup \tilde{m}_2$
$\tilde{n} \cap \tilde{m}_2 = \emptyset$
$\tilde{m}_1 \cap \tilde{m}_2 = \emptyset$

Definition 9. *Reduction in* SPiM.

The expression $Gillespie(V)$ chooses the next channel or delay on which to perform a reaction and calculates the duration t of the reaction, as described in Definition 8.

Finally, the expression $V \xrightarrow{r} V'$ simulates a single reaction for a machine term V and produces an updated machine term V', as described in Definition 9. The rate of the reaction is given by r, and the simulation time is incremented by the reaction time t. If a delay with rate r has been chosen from a term V by the Gillespie algorithm, and if the term contains a species with a delay $\tau_r.P$, the term can perform a reaction with rate r and then execute the process $\nu\tilde{m}\, P$ (9). If an interaction on channel x has been chosen from a term V by the Gillespie algorithm, and if the term contains a species with an output $!x(\tilde{n}).P_1$, together with a species with a corresponding input $?x(\tilde{m}).P_2$ then the input and output can interact on channel x with rate $\rho(x)$ and evolve to the process $\nu\tilde{m}_1\, \nu\tilde{m}_2\, (P_1 \mid P_{2\,\{\tilde{n}/\tilde{m}\}})$, where the value $\tilde{n}$ is bound to $\tilde{m}$ in P_2 (10).

4 Correctness

This section outlines a proof of correctness of the stochastic π-machine with respect to the stochastic π-calculus. Once the main technical lemmas and definitions have been formulated, the proofs themselves are relatively direct. .

The syntax of the stochastic π-calculus (SPi) is given in Definition 10, and is identical to the syntax described in [11]. The reduction rules of the calculus are given in Definition 11. In the general case, each rule is of the form $E \vdash P \xrightarrow{r} E \vdash P'$, which states that a system $E \vdash P$ can reduce to a system $E \vdash P'$ by doing a reaction with rate r. Since the environment E remains constant over time, the rules can be abbreviated to the form $P \xrightarrow{r} P'$. The reduction rules rely on a structural congruence relation, given in Definition 12, which defines a notion of equality on processes.

In this setting, the probability of performing the reaction $P \xrightarrow{r} P'$ is given by $r/R(P)$, where $R(P)$ denotes the apparent rate of P. This corresponds to the sum of the rates of all the reactions in P, and is defined as

$$R(P) \triangleq \sum_{\theta \in P} R(\theta, P) \tag{38}$$

for all the delays and channels in P, where θ can be a delay r or a channel x. By definition, $R(\theta, P)$ is the apparent rate of θ in process P, as described in Definition 13. The apparent rate of a given channel x is equal to the number of possible combinations of inputs and outputs on x, multiplied by the rate of x (34). The functions $\mathrm{In}_x(P)$ and $\mathrm{Out}_x(P)$ return the number of unguarded inputs and outputs on channel x in P, respectively, while $\mathrm{Mix}_x(P)$ returns the sum of $\mathrm{In}_x(M_i) \times \mathrm{Out}_x(M_i)$ for each choice M_i in P. The definition of apparent rate takes into account the fact that an input and an output in the same choice cannot interact, by subtracting $\mathrm{Mix}_x(P)$ from the product of the number of inputs and outputs on x. The apparent rate of a delay r is equal to the rate of the delay times the number of unguarded delays of rate r in P, written $\mathrm{Delay}_r(P)$ (35).

$P, Q ::=$	M	Choice	$E ::=$	$\emptyset$	Empty
$\mid$	$X(\tilde{n})$	Instance	$\mid$	$E, X(\tilde{m}) = P$	Definition
$\mid$	$P \mid Q$	Parallel			$\mathrm{fn}(P) \subseteq \tilde{m}$
$\mid$	$\nu x\, P$	Restriction			
			$\pi ::=$	$?x(\tilde{m})$	Input
$M ::=$	$\mathbf{0}$	Null	$\mid$	$!x(\tilde{n})$	Output
$\mid$	$\pi.P + M$	Action	$\mid$	τ_r	Delay

Definition 10. *Syntax of* SPi*, as defined in* [11].

$$\tau_r.P + M \xrightarrow{r} P \tag{11}$$

$$!x(\tilde{n}).P + M \mid ?x(\tilde{m}).Q + N \xrightarrow{\rho(x)} P \mid Q_{\{\tilde{n}/\tilde{m}\}} \tag{12}$$

$$P \xrightarrow{r} P' \quad \Rightarrow \quad \nu x\, P \xrightarrow{r} \nu x\, P' \tag{13}$$

$$P \xrightarrow{r} P' \quad \Rightarrow \quad P \mid Q \xrightarrow{r} P' \mid Q \tag{14}$$

$$Q \equiv P \xrightarrow{r} P' \equiv Q' \quad \Rightarrow \quad Q \xrightarrow{r} Q' \tag{15}$$

Definition 11. *Reduction in* SPi.

$$P \mid \mathbf{0} \equiv P \tag{16}$$

$$P \mid Q \equiv Q \mid P \tag{17}$$

$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R \tag{18}$$

$$X(\tilde{n}) \equiv P_{\{\tilde{n}/\tilde{m}\}} \textbf{ if } X(\tilde{m}) = P \tag{19}$$

$$\nu x\, \mathbf{0} \equiv \mathbf{0} \tag{20}$$

$$\nu x\, \nu y\, P \equiv \nu y\, \nu x\, P \tag{21}$$

$$\nu x\, (P \mid Q) \equiv P \mid \nu x\, Q \textbf{ if } x \notin \mathrm{fn}(P) \tag{22}$$

Definition 12. *Structural Congruence Axioms in* SPi. Structural congruence is defined as the least congruence that satisfies these axioms. Processes in SPi are also equal up to renaming of bound names and reordering of terms in a choice, as in [8].

The apparent rate $R(V)$ of a machine term V can be defined in a similar fashion, where $R(\theta, V)$ denotes the apparent rate of θ in term V, as described in Definition 14. The apparent rate of an unrestricted channel x is equal to the apparent rate of x in the heap (36). The apparent rate of a delay r is equal to the apparent rate of r in the heap, plus the apparent rates of all the restricted channels of rate r (37). The apparent rate of a restricted channel is recorded as a delay in order to ensure that the machine preserves the compositionality properties of the calculus. This is useful in cases where multiple machines are executed in parallel, for example in distributed or multi-core systems. In this setting, the restricted channels of a given machine are not be visible outside the scope of the machine, and interactions on these channels appear externally as delays.

The function $(\!|E \vdash P|\!)$ encodes a system $E \vdash P$ in SPi to a corresponding system in SPiM, as described in Definition 17. A corresponding decoding from the stochastic π-machine to the stochastic π-calculus is described in Definition 18.

$$\mathrm{In}_x(\nu x\, P) \triangleq 0 \tag{23}$$
$$\mathrm{In}_x(\nu y\, P) \triangleq \mathrm{In}_x(P) \textbf{ if } x \neq y \tag{24}$$
$$\mathrm{In}_x(P \mid Q) \triangleq \mathrm{In}_x(P) + \mathrm{In}_x(Q) \tag{25}$$
$$\mathrm{In}_x(X(\tilde{n})) \triangleq \mathrm{In}_x(P_{\{\tilde{n}/\tilde{m}\}}) \textbf{ if } X(\tilde{m}) = P \tag{26}$$

$$\mathrm{Delay}_r(\nu x\, P) \triangleq \mathrm{Delay}_r(P) \textbf{ if } \rho(x) \neq r \tag{27}$$
$$\mathrm{Delay}_r(\nu x\, P) \triangleq \mathrm{Delay}_r(P) + \mathrm{Act}_x(P) \textbf{ if } \rho(x) = r \tag{28}$$
$$\mathrm{Delay}_r(P \mid Q) \triangleq \mathrm{Delay}_r(P) + \mathrm{Delay}_r(Q) \tag{29}$$
$$\mathrm{Delay}_r(X(\tilde{n})) \triangleq \mathrm{Delay}_r(P_{\{\tilde{n}/\tilde{m}\}}) \textbf{ if } X(\tilde{m}) = P \tag{30}$$

$$\mathrm{Mix}_x(M) \triangleq \mathrm{In}_x(M) \times \mathrm{Out}_x(M) \tag{31}$$
$$\mathrm{Act}_x(P) \triangleq \mathrm{In}_x(P) \times \mathrm{Out}_x(P) - \mathrm{Mix}_x(P) \tag{32}$$
$$\tag{33}$$
$$R(x, P) \triangleq \rho(x) \times (\mathrm{Act}_x(P)) \tag{34}$$
$$R(r, P) \triangleq r \times \mathrm{Delay}_r(P) \tag{35}$$

Definition 13. *Apparent Rate in* SPi *based on [11].* The definitions of $\mathrm{In}_x(P)$, $\mathrm{Out}_x(P)$ and $\mathrm{Delay}_x(P)$ are given for processes P in SPi and extend the definitions of $\mathrm{In}_x(M)$, $\mathrm{Out}_x(M)$ and $\mathrm{Delay}_r(M)$ from Definition 4. The definitions of $\mathrm{Mix}_x(P)$ and $\mathrm{Out}_x(P)$ are similar to that of $\mathrm{In}_x(P)$ and are omitted.

$$R(x, (Z, H, S)) \triangleq a \textbf{ if } H(x) = (i, o, m, a) \textbf{ and } x \notin Z \tag{36}$$
$$R(r, (Z, H, S)) \triangleq a + \Sigma_i a_i \textbf{ if } H(r) = (d, a) \textbf{ and } x_i \in Z \tag{37}$$
$$\textbf{and } \rho(x_i) = r \textbf{ and } H(x_i) = (j_i, o_i, m_i, a_i)$$

Definition 14. *Apparent Rate in* SPiM.

Theorem 1 ensures that the terms of the stochastic π-machine are closed under reduction.

Theorem 1. $\forall E, V \in \mathrm{SPiM}.\ E \vdash V \xrightarrow{r} E \vdash V' \Rightarrow E \vdash V' \in \mathrm{SPiM}$

Proof. By induction on the derivation of reduction in SPiM □

Theorem 2 and Theorem 3 ensure that the stochastic π-calculus and the stochastic π-machine are reduction equivalent.

Theorem 2. $\forall E, V \in \mathrm{SPiM}.\ E \vdash V \xrightarrow{r} E \vdash V' \Rightarrow [\![E \vdash V]\!] \xrightarrow{r} [\![E \vdash V']\!]$

Proof. By induction on the derivation of reduction in SPiM □

Theorem 3. $\forall E, P \in \mathrm{SPi}.\ E \vdash P \xrightarrow{r} E \vdash P' \Rightarrow (\!|E \vdash P|\!) \xrightarrow{r} \equiv (\!|E \vdash P'|\!)$

$$(\!|\emptyset|\!) \triangleq \emptyset \tag{39}$$
$$(\!|E, X(\tilde{m}) = D|\!) \triangleq (\!|E|\!) \cup (\!|X(\tilde{m}) = D|\!) \tag{40}$$
$$(\!|X(\tilde{m}) = \nu\tilde{n} \textstyle\sum_{i=1}^{N} \pi_i.P_i|\!) \triangleq \bigcup_{i=1}^{N} E_i, X(\tilde{m}) = \nu\tilde{n} \textstyle\sum_{i=1}^{N} \pi_i.P_i' \textbf{ if } E_i \vdash P_i' = (\!|P_i|\!) \tag{41}$$
$$(\!|X(\tilde{m}) = P|\!) \triangleq E', X(\tilde{m}) = P' \textbf{ if } E' \vdash P' = (\!|P|\!) \text{ and } P \neq C \tag{42}$$

Definition 15. *Encoding an environment from* SPi *to* SPiM*, based on* [11]. The notation $\sum_{i=1}^{N} \pi_i.P_i$ is an abbreviation for a choice between zero or more actions $\pi_1.P_1 + \ldots + \pi_N.P_N + \mathbf{0}$.

$$(\!|\mathbf{0}|\!) \triangleq \emptyset \vdash \mathbf{0} \tag{43}$$
$$(\!|\nu\tilde{n}\, M|\!) \triangleq (\!|X(\tilde{m}) = \nu\tilde{n}\, M|\!) \vdash X(\tilde{m}) \textbf{ if } \tilde{m} = \mathrm{fn}(\nu\tilde{n}\, M) \textbf{ and } M \neq \mathbf{0} \textbf{ and} X \text{ fresh} \tag{44}$$
$$(\!|X(\tilde{n})|\!) \triangleq \emptyset \vdash X(\tilde{n}) \tag{45}$$
$$(\!|P_1 \mid P_2|\!) \triangleq E_1 \cup E_2 \vdash P_1' \mid P_2' \textbf{ if } E_1 \vdash P_1' = (\!|P_1|\!) \textbf{ and } E_2 \vdash P_2' = (\!|P_2|\!) \tag{46}$$
$$(\!|\nu x\, P|\!) \triangleq E \vdash \nu x\, P' \textbf{ if } E \vdash P' = (\!|P|\!) \textbf{ and } P \neq \nu\tilde{n}\, M \tag{47}$$

Definition 16. *Encoding a process from* SPi *to* SPiM*, based on* [11].

$$(\!|E \vdash P|\!) \triangleq (\!|E|\!) \cup E' \vdash (\emptyset, \emptyset, \emptyset) \oplus P' \textbf{ if } E' \vdash P' = (\!|P|\!) \tag{48}$$

Definition 17. *Encoding a system from* SPi *to* SPiM.

$$[\![E \vdash V]\!] \triangleq E \vdash [\![V]\!] \tag{49}$$
$$[\![Z, S, H]\!] \triangleq \nu Z\, [\![H]\!] \tag{50}$$
$$[\![\emptyset]\!] \triangleq \mathbf{0} \tag{51}$$
$$[\![H, X(\tilde{n}) \mapsto (i, U, C)]\!] \triangleq \underbrace{X(\tilde{n}) \mid \ldots \mid X(\tilde{n})}_{i} \mid [\![H]\!] \tag{52}$$

Definition 18. *Decoding a system from* SPiM *to* SPi. The environment E is unchanged (49), and for each mapping $X(\tilde{n}) \mapsto (i, U, C)$ in the heap, i copies of the instance are executed in parallel (52).

Proof. By induction on the derivation of reduction in SPi, where machine terms are structurally congruent up to renaming of definitions, garbage-collection of unused definitions and structural congruence of processes. These assumptions are necessary since the definitions created in the encoding $(\!|E \vdash P|\!)$ can have different names to those created in $(\!|E \vdash P'|\!)$. Similarly, $(\!|E \vdash P'|\!)$ can have less definitions than $(\!|E \vdash P|\!)$ after the process P has been reduced. □

Finally, Theorem 4 and Theorem 5 ensure that the apparent rate of reactions is preserved by encoding and decoding.

Theorem 4. $\forall P, \theta \in \text{SPi}. R(\theta, P) = R(\theta, (\!|P|\!))$

Proof. By induction on the derivation of encoding in SPi □

Theorem 5. $\forall V, \theta \in \text{SPiM}. R(\theta, V) = R(\theta, [\![V]\!])$

Proof. By induction on the derivation of decoding in SPiM □

5 Implementation

This section shows how the simulation algorithm of Sec. 3 can be mapped to functional program code, in order to implement a stochastic simulator. The mapping is relatively direct, indicating that the algorithm is sufficiently low-level to be readily implemented.

The processes of the stochastic π-machine are implemented as functional datatypes, as shown in Fig. 2. In addition, the environment, the store and the heap are implemented using a standard map library, where StringMap, SpeciesMap and ValueMap are maps indexed by strings X, species I and values θ, respectively, and a value can be a delay r or a channel x. A term is implemented as a triple consisting of a counter, a store and a heap. Each time a fresh channel is created, the counter is incremented and used to generate a fresh name. As a result, the term does not need to explicitly store all the private channels in the system, since the counter keeps track of all the channels that have been created, thereby preventing name clashes.

The implementation of reduction is also described in Fig. 2. The function *reduce* is based on Definition 9 while the function *add* is based on Definition 7. The function *remove* is based on equation (3) of Definition 6, and is implemented so that each delay of a given rate r or interaction on a given channel x has an equal probability of being selected, once a particular delay rate or interaction channel has been chosen by the Gillespie algorithm. The new simulator has been tested on the full range of examples available from [9], in most cases with significant improvement in efficiency. For instance, the example from Sec. 2 with 100 copies of each gene and simulation time 200000 took 8 minutes in the previous version of SPiM, but just 10 seconds in the optimised version (compared with 21 minutes in the BioSPI simulator). As a first step this paper focuses on reducing the algorithmic complexity of the simulation algorithm, rather than optimising the final implementation.

In the previous version of the simulator, a separate process is created for each gene or protein in the system, as described in [10]. In terms of efficiency, this is analogous to defining the heap H of a machine term as a list of choices $C_1 :: \ldots :: C_N$ instead of a mapping $I_1 \mapsto (i_1, U_1, C_1), \ldots, I_M \mapsto (i_M, U_M, C_M)$ to keep track of the number of copies of each choice. In both versions the cost of computing the Gillespie algorithm to choose the next reaction is unchanged. However, in the previous version the cost of finding a choice to execute a reaction is $O(N)$, where N is the number of choices, while the cost of inserting a choice is constant, since choices are inserted at the head of the list. In the new version the cost of finding a choice to execute a reaction is $O(M)$, where M is the number of species, while the cost of insertion is $O(\log\ M)$, assuming that the heap is a balanced tree. In addition, the new version pre-computes the number of inputs, outputs and delays for each species in the substore U, so that the store S can be

```
type action =
    Input of value * pattern
  | Output of value * value
  | Delay of value

type process =
    Null
  | Instance of string * value
  | Parallel of process * process
  | New of value * process

type choice =
  value * ((action * process) list)

type definition =
    Process of process
  | Choice of choice

type env = (pattern * definition) StringMap.t
type record = (int * int * int * int * float)
type store = record ValueMap.t
type heap = (int * store * choice) SpeciesMap.t
type term = int * store * heap

let reduce (e:env) (t:term) = match gillespie t with
    None -> None
  | Some(Rate(r),time) -> ( match remove (Delay(r)) t with
        Some(m,Delay(r),p,t') -> Some(time,add e (New(m,p)) t')
      | _ -> None )
  | Some(Channel(x),time) -> match remove (Input(x,m0)) t with
        Some(m1,Input(x,m),p1,t') -> ( match remove (Output(x,v0)) t' with
              Some(m2,Output(x,n),p2,t') ->
                let p2 = bind (eval n) m p2
                in Some(time,add e (New(m1,New(m2,Parallel(p1,p2)))) t')
            | _ -> None )
      | _ -> None
```

Fig. 2. Implementing SPiM in OCaml

quickly updated whenever there is a change in the species population. Inferring a species name for each choice also allows various additional optimisations to be implemented. For example, the current version of SPiM keeps a lookup table inside U to track which species can input and output on which channels, allowing the cost of finding a species to be further reduced to a lookup $O(\log\ M)$. In situations where each species has a population of 1 there will be little improvement, apart from not having to re-compute the number delays, inputs and outputs for each species. However, in situations where the population of species is large, which is very common in a biological setting, there will be significant improvement. For example, in the system of Fig. 1 there are generally thousands of copies of a given protein.

6 Conclusions

This paper presented a simulation algorithm for the stochastic π-calculus, designed for the efficient simulation of biological systems with large numbers of

molecules. The algorithm was proved correct with respect to the calculus, and then used as the basis for implementing an efficient simulator. To our knowledge, this is the first provably correct simulation algorithm for the stochastic π-calculus to formalise such optimisations.

Previous simulators for the stochastic π-calculus include the BioSPI simulator [1], the StoPi simulator [13], and an earlier version of the SPiM simulator [9]. The main difference with the current work is that these simulators do not formally describe an algorithm for keeping track of identical processes. The simulation algorithm of [10] was proved correct with respect to a variant of the stochastic π-calculus, and then mapped to executable program code in order to implement a stochastic simulator. This paper uses similar techniques, applied to a more efficient algorithm. In more recent work [11], a graphical variant of the stochastic π-calculus was presented, together with a corresponding graphical execution model. The graphical calculus required each choice to be associated with a corresponding identifier, so that it could be traced during execution. This paper uses similar syntactic constraints, allowing a graphical representation to be generated after each reaction as shown in Fig. 1, by adapting the graph generation algorithm of [11].

There are a number of improvement in this paper with respect to the original algorithm in [10]. In addition to the syntactic constraint placed on calculus processes, choices are dynamically grouped into species during execution according to their identifier and associated parameters. The algorithm also introduces store and substore data structures, which keep track of all the possible reactions in the heap and in the individual species, respectively. The corresponding correctness proof takes into account these extensions by checking that the store and substore data structures remain consistent with the heap, and by ensuring that the syntactic constraints do not introduce simulation errors.

There are a number of areas of future work. In the short term, the prototype simulator presented in this paper will form the basis of the next release of the Stochastic Pi Machine, available from [9]. The algorithm presented in this paper is also being extended in order to efficiently handle the dynamic creation of complexes during a simulation. Preliminary results indicate that a suitable extension can be defined with relatively few changes to the existing machine. The algorithm presented in this paper exploits the fact that biological systems typically contain large numbers of processes with identical behaviour, in contrast with most computer systems. In future, more specific optimisations for the algorithm could be investigated, such as the use of more refined data structures like priority queues, in the style of [4]. There also seems to be a close link between models that can be efficiently simulated, and those that are amenable to formal analysis, since the size of the model needs to be reduced in both cases. More generally, the simulation algorithm presented in this paper has a broader scope beyond the stochastic π-calculus, and could in principle be applied to a range of name-passing process calculi for biological modelling such as [14], in order to develop efficient simulators that are provably correct.

References

1. Bloch, A., Haagensen, B., Hoyer, M.K., Knudsen, S.U.: The StoPi-calculus and Simulator, http://www.cs.aau.dk/bh/education.html
2. Blossey, R., Cardelli, L., Phillips, A.: A compositional approach to the stochastic dynamics of gene networks. Transactions in Computational Systems Biology 3939, 99–122 (2006)
3. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Software-Practice and Experience , 1–5 (1999)
4. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. 104, 1876–1889 (2000)
5. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81(25), 2340–2361 (1977)
6. Gillespie, D.T.: Approximate accelerated stochastic simulation of chemically reacting systems. J. Chem. Phys. 115, 1716–1733 (2001)
7. Lecca, P., Priami, C.: Cell cycle control in eukaryotes: a biospi model. In: BioConcur'03. ENTCS (2003)
8. Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press, Cambridge (1999)
9. Phillips, A.: The Stochastic Pi-Machine (2006), Available from http://research.microsoft.com/~aphillip/spim/
10. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. In: Bioconcur'04, ENTCS (August 2004)
11. Phillips, A., Cardelli, L., Castagna, G.: A graphical representation for biological processes in the stochastic pi-calculus. Transactions in Computational Systems Biology 4230, 123–152 (2006)
12. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters 80, 25–31 (2001)
13. Regev, A., Silverman, W., Shapiro, E.: Representation and simulation of biochemical processes using the pi- calculus process algebra. In: Pacific Symposium on Biocomputing, vol. 6, pp. 459–470 (2001)
14. Romanel, A., Dematte, L., Priami, C.: The Beta Workbench. Available from, http://www.cosbi.eu/Rpty_Soft_BetaWB.php
15. Tian, T., Burrage, K.: Binomial leap methods for simulating stochastic chemical kinetics. J. Chem. Phys. 121, 10356–10364 (2004)
16. David, N.: Turner. The Polymorphic Pi-Calculus: Theory and Implementation. PhD thesis, June, CST-126-96 (also published as ECS-LFCS-96-345) (1996)

A Unifying Framework for Modelling and Analysing Biochemical Pathways Using Petri Nets

David Gilbert[1], Monika Heiner[2], and Sebastian Lehrack[3]

[1] Bioinformatics Research Centre, University of Glasgow
Glasgow G12 8QQ, Scotland, UK
drg@brc.dcs.gla.ac.uk, on sabbatical leave from [3]
[2] INRIA Rocquencourt, Projet Contraintes
BP 105, 78153 Le Chesnay CEDEX - France
monika.heiner@inria.fr, on sabbatical leave from [3]
[3] Department of Computer Science, Brandenburg University of Technology
Postbox 10 13 44, 03013 Cottbus, Germany
sebastian.lehrack@informatik.tu-cottbus.de

Abstract. We give a description of a Petri net-based framework for modelling and analysing biochemical pathways, which unifies the qualitative, stochastic and continuous paradigms. Each perspective adds its contribution to the understanding of the system, thus the three approaches do not compete, but complement each other. We illustrate our approach by applying it to an extended model of the three stage cascade, which forms the core of the ERK signal transduction pathway. Consequently our focus is on transient behaviour analysis. We demonstrate how qualitative descriptions are abstractions over stochastic or continuous descriptions, and show that the stochastic and continuous models approximate each other. A key contribution of the paper consists in a precise definition of biochemically interpreted stochastic Petri nets. Although our framework is based on Petri nets, it can be applied more widely to other formalisms which are used to model and analyse biochemical networks.

1 Motivation

Biochemical systems are inherently governed by stochastic laws. However, due to the computational efforts required to analyse stochastic models, two abstractions are more popular: qualitative models, abstracting away from any time dependencies, and continuous models, commonly used to approximate stochastic behaviour by a deterministic one. The interrelationships between these three models are not always properly understood; for example, how the kinetics of a biochemical reaction, when described by a continuous model, is related to the stochastic nature of the underlying molecular mechanism.

In a previous paper [GH06] we developed an approach for modelling and analysing biochemical networks using discrete and continuous Petri nets. Our current work has taken this forward by considering stochastic Petri nets and

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 200–216, 2007.

developing an overall framework to unify these three approaches, providing a family of related models with high analytical power.

A key contribution of this paper is the precise definition of biochemically interpreted stochastic Petri nets in a generic manner, and we demonstrate the benefit of their incorporation into the model development process. We show how the general definition can be tailored to very specific kinetic assumptions by appropriate adjustments of the general hazard function. Also we discuss the relation of the stochastic Petri net to its time-free, purely qualitative abstraction - the standard Petri net, as well as to its continuous approximation - the continuous Petri net (i.e., an ordinary differential equation system).

This paper is organised as follows. The following section provides an overview of the biochemical context and introduces our running example. Next we outline our framework, discussing the special contributions of the three individual analysis approaches with special emphasis on the transient behaviour analysis, and examining their interrelations. We then present the individual approaches and discuss mutually related properties in all three paradigms in the following order: we start off with the qualitative approach, which is conceptually the easiest, and does not rely on knowledge of kinetic information, but describes the network topology and presence of the species. The qualitative modelling and analysis basically adheres to the steps proposed in [GH06]. In addition, we show how to systematically derive and interpret the partial order run of the signal response behaviour. We then demonstrate how the validated qualitative model can be transformed into the stochastic representation by addition of stochastic firing rate information. Next, the continuous model is derived from the qualitative or stochastic model by considering only deterministic firing rates. Suitable sets of initial conditions for all three models are constructed by qualitative analysis. We conclude with a summary and outlook regarding further research directions.

2 Biochemical Context

We have chosen a model of the mitogen-activated protein kinase (MAPK) cascade published in [LBS00] as a running case study. This is the core of the ubiquitous ERK/MAPK pathway that can, for example, convey cell division and differentiation signals from the cell membrane to the nucleus. The model does not describe the receptor and the biochemical entities and actions immediately downstream from the receptor. Instead the description starts at the RasGTP complex which acts as a kinase to phosphorylate Raf, which phosphorylates MAPK/ERK Kinase (MEK), which in turn phosphorylates Extracellular signal Regulated Kinase (ERK). This cascade (RasGTP $\rightarrow$ Raf $\rightarrow$ MEK $\rightarrow$ ERK) of protein interactions controls cell differentiation, the effect being dependent upon the activity of ERK. We consider RasGTP as the input signal and ERKPP (activated ERK) as the output signal.

The bipartite graph in Figure 1 describes the typical modular structure for such a signalling cascade. Each layer corresponds to a distinct protein species. The protein Raf in the first layer is only singly phosphorylated. The proteins

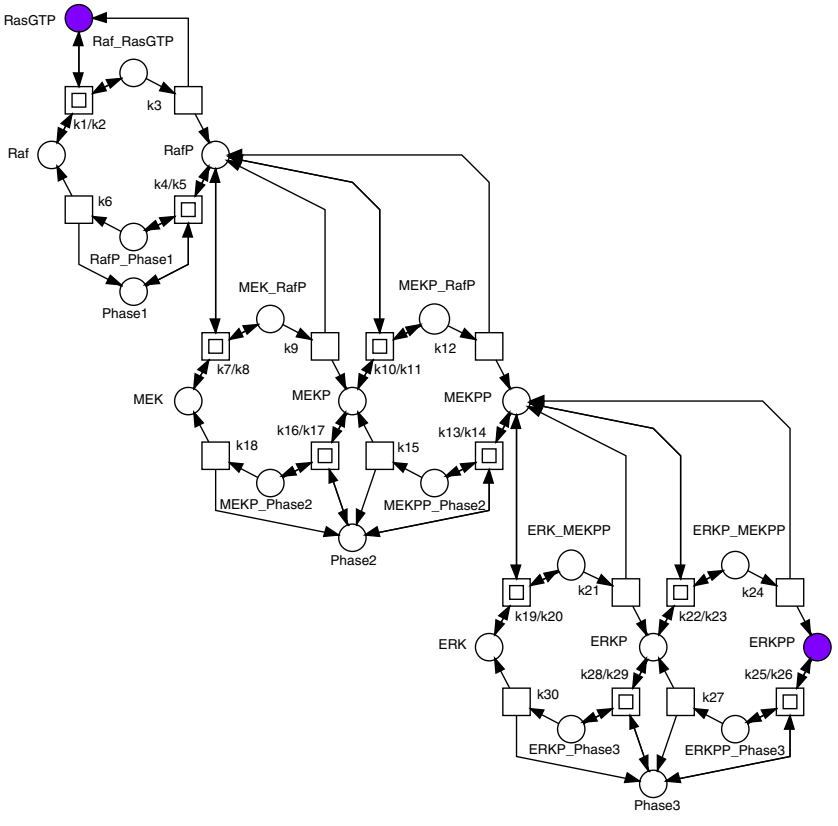


Fig. 1. The bipartite graph for the extended ERK pathway model. The graph has been derived by SBML import and automatic layout, manually improved, from the set of the ODEs in [LBS00]. Circles stand for species (proteins, protein complexes). Protein complexes are indicated by an underscore "_" between the constituent protein names. The suffixes P or PP indicate phosphorylated or doubly phosphorylated forms respectively. Squares stand for irreversible reactions, while two concentric squares specify reversible reactions. The species that are read as input/output signals are given in grey.

in the two other layers, MEK and ERK respectively, can be singly as well as doubly phosphorylated. In each layer, forward reactions are catalysed by kinases and reverse reactions by phosphatases (Phase1, Phase2, Phase3). The kinases in the MEK and ERK layers are the phosphorylated forms of the proteins in the previous layer, see also [CKS07].

3 Overview of the Framework

In the following we describe our overall framework, illustrated in Figure 2, that relates the three major ways of modelling and analysing biochemical networks described in this paper: qualitative, stochastic and continuous.

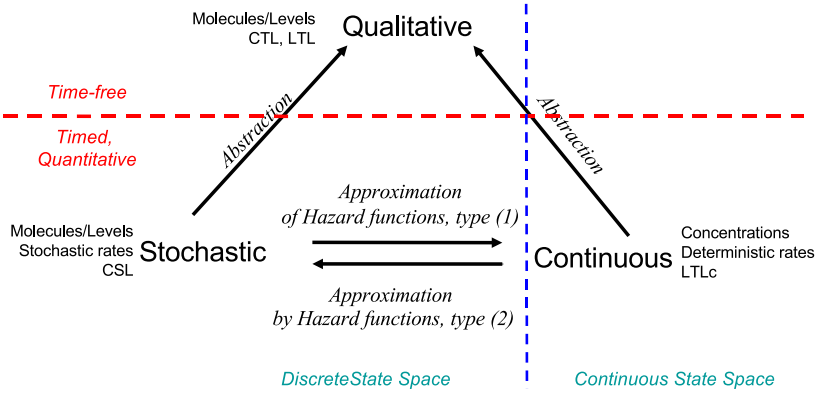


Fig. 2. Conceptual framework

The most abstract representation of a biochemical network is *qualitative* and is minimally described by its topology, usually as a bipartite directed graph with nodes representing biochemical entities or reactions, or in Petri net terminology *places* and *transitions* (see Figure 1). Arcs can be annotated with stoichiometric information, whereby the default stoichiometric value of 1 is usually omitted.

The qualitative description can be further enhanced by the abstract representation of discrete quantities of species, achieved in Petri nets by the use of tokens at places. These can represent the number of molecules, or the level of concentration, of a species, and a particular arrangement of tokens over a network is called a *marking*. The standard semantics for these qualitative Petri nets (QPN) does not associate a time with transitions or the sojourn of tokens at places, and thus these descriptions are time-free. The qualitative analysis considers however all possible behaviour of the system under any timing. The behaviour of such a net forms a discrete state space, which can be analysed in the bounded case, for example, by a branching time temporal logic, one instance of which is Computational Tree Logic (CTL), see [CGP01].

Timed information can be added to the qualitative description in two ways – stochastic and continuous. The stochastic Petri net (SPN) description preserves the discrete state description, but in addition associates a probabilistically distributed firing rate (waiting time) with each reaction. All reactions, which occur in the QPN, can still occur in the SPN, but their likelihood depends on the probability distribution of the associated firing rates (waiting times). Special behavioural properties can be expressed using e.g. Continuous Stochastic Logic (CSL), see [PNK06], a probabilistic counterpart of CTL. The QPN is an abstraction of the SPN, sharing the same state space and transition relation with the stochastic model, with the probabilistic information removed. All qualitative properties valid in the QPN are also valid in the SPN, and vice versa.

The continuous model replaces the discrete values of species with continuous values, and hence is not able to describe the behaviour of species at the level

of individual molecules, but only the overall behaviour via concentrations. We can regard the discrete description of concentration levels as abstracting over the continuous description of concentrations. Timed information is introduced by the association of a particular deterministic rate information with each transition, permitting the continuous model to be represented as a set of ordinary differential equations (ODEs). The concentration of a particular species in such a model will have the same value at each point of time for repeated experiments. The state space of such models is continuous and linear. It can be analysed by, for example, Linear Temporal Logic with constraints (LTLc) in the manner of [CCRFS06].

The stochastic and continuous models are mutually related by approximation. The stochastic description can be used as the basis for deriving a continuous Petri net (CPN) model by approximating rate information. Specifically, the probabilistically distributed reaction firing in the SPN is replaced by a particular average firing rate over the continuous token flow of the CPN. This is achieved by approximation over hazard functions of type (1), described in more detail in section 5.1. In turn, the stochastic model can be derived from the continuous model by approximation, reading the tokens as concentration levels, as introduced in [CVGO06]. Formally, this is achieved by a hazard function of type (2), see again section 5.1.

It is well-known that time assumptions generally impose constraints on behaviour. The qualitative and stochastic models consider all possible behaviours under any timing, whereas the continuous model is constrained by its inherent determinism to consider a subset. This may be too restrictive when modelling biochemical systems, which by their very nature exhibit variability in their behaviour.

In the following the reader is assumed to be familiar with the standard Petri net terminology as well as foundations of temporal logics, for an introduction see, e.g., [Mur89] and [CGP01].

4 The Qualitative Approach

4.1 Qualitative Modelling

We interpret the graph given in Figure 1 as a place/transition Petri net, and call the circles *places*, and the rectangles *transitions*. Reversible reactions have to be modelled explicitly by two opposite transitions in the basic Petri net notation. However in order to retain the elegant graph structure of Figure 1, we use macro transitions, each of which stands here for a reversible reaction. The entire (flattened) place/transition Petri net consists of 22 places and 30 transitions, where $k1, k2, \ldots$ stand for reaction (transition) labels.

We associate a discrete concentration with each of the 22 species. In the simplest way, these concentrations can be thought of as being “high” or “low” (above or below a certain threshold), resulting in a two-level model where each species can be read as a Boolean variable. More generally, we could apply a multi-level approach by differentiating between a finite number of discrete levels,

each standing for an equivalence class of possibly infinitely many concentrations. Then species can be read as integer variables.

4.2 Qualitative Analysis

Analysis of general behavioural properties. The Petri net enjoys the three orthogonal general properties of a discrete Petri net: boundedness, liveness and reversibility. The decision about the first two can be made for our running example in a static way, while the last property requires dynamic analysis techniques. The necessary steps of the systematic analysis procedure follow basically those given in [GH06]. We restrict ourselves here to the most essential points.

The net is *strongly connected* and thus self-contained, i.e. a closed system. In order to bring the net to life, we construct an initial marking using *P-invariants*. These are non-trivial non-negative integer solutions of the homogeneous linear equation system $x \cdot \mathcal{C} = 0$, where $\mathcal{C}$ stands for the incidence matrix of the net. There are seven minimal P-invariants covering the net, and consequently the net is bounded for any initial marking. All these P-invariants x_i contain only entries of 0 and 1, permitting a short-hand specification by just giving the names of the places involved.

x_1 = (**RasGTP,** Raf_RasGTP)
x_2 = (**Raf,** Raf_RasGTP, RafP, RafP_Phase1, MEK_RafP, MEKP_RafP)
x_3 = (**MEK,** MEK_RafP, MEKP_RafP, MEKP_Phase2, MEKPP_Phase2, ERK_MEKPP, ERKP_MEKPP, MEKPP, MEKP)
x_4 = (**ERK,** ERK_MEKPP, ERKP_MEKPP, ERKP, ERKPP_Phase3, ERKP_Phase3, ERK_PP)
x_5 = (**Phase1,** RafP_Phase1)
x_6 = (**Phase2,** MEKP_Phase2, MEKPP_Phase2)
x_7 = (**Phase3,** ERKP_Phase3, ERKPP_Phase3)

Each P-invariant stands for a reasonable conservation rule, the species preserved being given by the first name in the invariant. In signal transduction networks a P-invariant typically comprises all the different states of one species. In a Boolean approach, each species can be only in one state at any time, thus each P-invariant gets exactly one token. Within a P-invariant, the species with the most *inactive* (i.e. non-phosphorylated) or the *monomeric* (non-complexed) state is chosen. Following these criteria, the initial marking is: one token on each of Raf, RasGTP, MEK, ERK, Phase1, Phase2 and Phase3, while all remaining places are empty. With this marking, the net is covered by 1-P-invariants (exactly one token in each P-invariant), and is therefore 1-bounded.

Generalising this reasoning to a multi-level concept, we could assign n tokens to each place, representing the most inactive state, in order to indicate the highest concentration level for them in the initial state. The "abstract" mass conservation within each P-invariant would then be n tokens, which could be distributed fairly freely over the P-invariant's places during the behaviour of the model. This results in a dramatic increase of the state space, cf. the discussion in Section 5.2, while not improving the qualitative reasoning.

Model validation should include a check of all *T-invariants* for their biological plausibility. T-invariants are non-trivial non-negative integer solutions of the homogeneous linear equation system $\mathcal{C} \cdot y = 0$. The entries of a T-invariant can be read as the specification of a multiset of transitions, which reproduce a given marking by their firing. If there are non-trivial solutions, then there are infinitely many ones. Therefore, the plausibility check is usually restricted to the consideration of all minimal solutions. The net representations of minimal T-invariants (their transitions plus their pre- and post-places and all arcs in between) characterise minimal self-contained subnetworks with an identifiable biological meaning.

The net under consideration is covered by T-invariants, a necessary condition for bounded nets to be live. Besides the expected ten trivial T-invariants for the ten reversible reactions, there are five non-trivial, but obvious minimal T-invariants, each corresponding to one of the five phosphorylation / dephosphorylation cycles in the network structure:

y_1 = (k1, k3, k4, k6), y_2 = (k7, k9, k16, k18), y_3 = (k10, k12, k13, k15),
y_4 = (k19, k21, k28, k30), y_5 = (k22, k24, k25, k27).

The interesting net behaviour, demonstrating how input signals cause finally output signals, is contained in a non-negative linear combination of all five non-trivial T-invariants, $y_{1-5} = y_1 + y_2 + y_3 + y_4 + y_5$, which is called an I/O T-invariant in the following. The I/O T-invariant is systematically constructed by starting with the two minimal T-invariants, involving the input and output signal, which define disconnected subnetworks. Then we add minimal sets of minimal T-invariants to get a connected subnet. For our running example, the solution is unique, which is not generally the case.

We check the I/O T-invariant for feasibility in the constructed initial marking, which then involves the feasibility of all trivial T-invariants. We obtain an *infinite partial order run*, the beginning of which can be characterised in a shorthand notation by the following partially ordered word out of the alphabet of all transition labels (";" stands for "sequentiality", "‖" for "concurrency"):

(k1; k3; k7; k9; k10; k12;
((k4; k6) ‖ ((k19; k21; k22; k24); ((k13; k15; k16; k18) ‖ (k25; k27; k28; k30))))),

see [GHL07] for a graphical representation. This partial order run gives further insight into the dynamic behaviour of the network, which may not be apparent from the standard net representation, e.g. we are able to follow the (minimal) producing process of the proteins RafP, MEKP, MEKPP, ERKP and ERKPP, compare [GHL07], and we notice the clear independence of the dephosphorylation in all three levels.

The *reachability graph* of the net is finite because the net is bounded, and has in the Boolean token interpretation 118 states out of 2^{22} theoretically possible ones, forming one strongly connected component. Therefore, the Petri net is reversible, i.e. the initial system state is always reachable again, or in other words the system has the capability of self-reinitialization. Moreover, from the

viewpoint of the discrete model, all of these 118 states are equivalent, and each could be taken as an initial state resulting in exactly the same total (discrete) system behaviour. This prediction will be confirmed by the observations gained during quantitative analyses, see Sections 5.2 and 6.2.

Model checking of special behavioural properties. Temporal logic is particularly helpful in expressing special behavioural properties of the expected transient behaviour, whose truth can be determined via model checking. We confine ourselves here to two CTL properties, checking the generalizability of the insights gained by the partial order run of the I/O T-invariant. In the following, places are interpreted as Boolean variables, in order to simplify notation.

Property Q1: The signal sequence predicted by the partial order run of the I/O T-invariant is the only possible one. In other words, starting at the initial state, it is necessary to pass through states RafP, MEKP, MEKPP and ERKP in order to reach ERKPP.

$$\neg\,[\ \mathbf{E}\ (\ \neg\ \text{RafP}\quad\ \mathbf{U}\ \text{MEKP}\ )\ \vee\ \mathbf{E}\ (\ \neg\ \text{MEKP}\ \ \mathbf{U}\ \text{MEKPP}\ )\ \vee\ \mathbf{E}\ (\ \neg\ \text{MEKPP}\ \ \mathbf{U}\ \ \text{ERKP}\ )\ \vee\ \mathbf{E}\ (\ \neg\ \text{ERKP}\ \ \mathbf{U}\ \ \text{ERKPP}\ )\ \ ]$$

Property Q2: Dephosphorylation takes place independently. E.g., the duration of the phosphorylated state of ERK is independent of the duration of the phosphorylated states of MEK and Raf.

$$(\ \mathbf{EF}\ [\ \text{Raf} \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ] \wedge \mathbf{EF}\ [\ \text{RafP} \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ] \wedge \mathbf{EF}\ [\ \text{MEK} \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ] \wedge \mathbf{EF}\ [\ (\ \text{MEKP} \vee \text{MEKPP}\ ) \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ]\ )$$

In subsequent sections we will use Q1 as a basis to illustrate how the stochastic and continuous approaches provide complementary views of the system behaviour.

5 The Stochastic Approach

5.1 Stochastic Modelling

As with a qualitative Petri net, a stochastic Petri net maintains a discrete number of tokens on its places. But contrary to the time-free case, a firing rate (waiting time) is associated with each transition t, which are random variables $X_t \in [0, \infty)$, defined by probability distributions. Therefore, all reaction times can theoretically still occur, but the likelihood depends on the probability distribution. Consequently, the system behaviour is described by the same discrete state space, and all the different execution runs of the underlying qualitative Petri net can still take place. This allows the use of the same powerful analysis techniques for stochastic Petri nets as already applied for qualitative Petri nets.

For a better understanding we describe the general procedure of a particular simulation run for a stochastic Petri net. Each transition gets its own local timer. When a particular transition becomes enabled, meaning that sufficient tokens arrive on its pre-places, then the local timer is set to an initial value,

which is computed at this time point by means of the corresponding probability distribution. In general, this value will be different for each simulation run. The local timer is then decremented at a constant speed, and the transition will fire when the timer reaches zero. If there is more than one enabled transition, a race for the next firing will take place.

Technically, various probability distributions can be chosen to determine the random values for the local timers. Biochemical systems are the prototype for exponentially distributed reactions. Thus, for our purposes, the firing rates of all transitions follow an exponential distribution, which can be described by a single parameter λ, and each transition needs only its particular, generally marking-dependent parameter λ to specify its local time behaviour.

Definition 1 (Stochastic Petri net, Syntax). *A* biochemically interpreted stochastic Petri net *is a quintuple* $\mathcal{SPN}_{Bio} = (P, T, f, v, m_0)$*, where*

- P *and* T *are finite, non empty, and disjoint sets.* P *is the set of* places*, and* T *is the set of* transitions*.*
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}_0$ *defines the set of directed* arcs*, weighted by non-negative integer values.*
- $v : T \to H$ *is a function, which assigns a* stochastic hazard function h_t *to each transition* t*, whereby* $H := \bigcup_{t \in T} \left\{ h_t \,|\, h_t : \mathbb{N}_0^{|{}^\bullet t|} \to \mathbb{R}^+ \right\}$ *is the set of all stochastic hazard functions, and* $v(t) = h_t$ *for all transitions* $t \in T$*.*
- $m_0 : P \to \mathbb{N}_0$ *gives the* initial marking*.*

The stochastic hazard function h_t defines the marking-dependent transition rate $\lambda_t(m)$ for the transition t. The domain of h_t is restricted to the set of pre-places of t, i.e. ${}^\bullet t := \{p \in P | f(p,t) \neq 0\}$, to enforce a close relation between network structure and hazard functions. Therefore $\lambda_t(m)$ actually depends only on a sub-marking.

Stochastic Petri net, Semantics. Transitions become enabled as usual, i.e. if all pre-places are sufficiently marked. However there is a time, which has to elapse, before an enabled transition $t \in T$ fires. The transition's waiting time is an exponentially distributed random variable X_t with the *probability density function*:

$$f_{X_t}(\tau) = \lambda_t(m) \cdot \mathrm{e}^{(-\lambda_t(m) \cdot \tau)}, \qquad \tau \geq 0.$$

The firing itself does not consume time and again follows the standard firing rule of qualitative Petri nets. The semantics of a stochastic Petri net (with exponentially distributed reaction times for all transitions) is described by a continuous time Markov chain (CTMC). The CTMC of a stochastic Petri net is isomorphic to the reachability graph of the underlying qualitative Petri net, while the arcs between the states are now labelled by the transition rates. For more details see [Mur89], [BK02].

Based on this general $\mathcal{SPN}_{Bio}$ definition, specialised biochemically interpreted stochastic Petri nets can be defined by specifying the required kind of

stochastic hazard function more precisely. We give two examples, reading the tokens as molecules or as concentration levels. The *stochastic mass-action hazard function* tailors the general $\mathcal{SPN}_{Bio}$ definition to biochemical mass-action networks, where tokens correspond to molecules:

$$h_t := c_t \cdot \prod_{p \in {}^\bullet t} \binom{m(p)}{f(p,t)}, \tag{1}$$

where c_t is the transition-specific stochastic rate constant, and $m(p)$ is the current number of tokens on the pre-place p of transition t. The binomial coefficient describes the number of non-ordered combinations of the $f(p,t)$ molecules, required for the reaction, out of the $m(p)$ available ones.

Tokens can also be read as concentration levels, as introduced in [CVGO06]. The current concentration of each species is given as an abstract level. We assume the maximum molar concentration is M, and the amount of different levels is $N+1$. Then the abstract values $0, \ldots, N$ represent the concentration intervals $0,\ (0,\ 1*M/N],\ (1*M/N,\ 2*M/N],\ \ldots,\ (N-1*M/N,\ N*M/N]$. Each of these (finite many) discrete levels stands for an equivalence class of (infinitely many) continuous states. The *stochastic level hazard function* tailors the general $\mathcal{SPN}_{Bio}$ definition to biochemical mass-action networks, where tokens correspond to concentration levels:

$$h_t := k_t \cdot N \cdot \prod_{p \in {}^\bullet t} \left(\frac{m(p)}{N}\right), \tag{2}$$

where k_t is the transition-specific deterministic rate constant, and N the number of the highest level. The transformation rules between the stochastic and deterministic rate constants are well-understood, see e.g. [Wil06]. In practice, kinetic rates are taken from literature, textbooks etc. or determined from biochemical experiments. Hazard function (2) is the means whereby the continuous model (see the framework in Figure 2 and Section 6) can be approximated by the stochastic model; this can generally be achieved by a limited number of levels – see Section 5.2.

5.2 Stochastic Analysis

Due to the isomorphy of the reachability graph and the CTMC, all qualitative analysis results obtained in Section 4 are still valid. The influence of time does not restrict the possible system behaviour. Specifically it holds that the CTMC of our case study is reversible, which ensures ergodicity; i.e. we could start the system in any of the reachable states, always resulting in the same CTMC with the same steady state probability distribution.

In the following our main focus is on the analytic model checking approach. In Section 4.2 we employed CTL to express behavioural properties. Since we have now a stochastic model, we apply Continuous Stochastic Logic (CSL) [PNK06], which replaces the path quantifiers (**E**, **A**) in CTL by the probability operator

$\mathbf{P}_{\bowtie p}$, whereby $\bowtie p$ specifies the probability of the given formula. For example, introducing in CSL the abbreviation $\mathbf{F}\,\phi$ for $true\,\mathbf{U}\,\phi$, the CTL formula $\mathbf{EF}\,\phi$ becomes the CSL formula $\mathbf{P}_{\geq 0}[\,\mathbf{F}\,\phi\,]$, and $\mathbf{AF}\,\phi$ becomes $\mathbf{P}_{\geq 1}[\,\mathbf{F}\,\phi\,]$.

In order to use the probabilistic model checker PRISM [PNK06], we encode the extended ERK pathway in its modelling language, as proposed in [DDS04]. This translation requires knowledge of the boundedness degree of all species involved, which we acquire by the structural analysis technique of P-invariants.

We only consider here the level semantics. Since the continuous concentrations of proteins in the ERK pathway are all in the same range (0.1...0.4 mMol in 0.1 steps), we employ a model with only 4, and a second version with 8 levels. The corresponding CTMCs (and reachability graphs) comprise 24,065 states for the 4 level version and 6,110,643 states for the 8 level version.

Equivalence check by transient analysis. We start with a transient analysis to prove the sufficient equivalence between the stochastic model in the level semantics and the corresponding continuous model, justifying the interpretation of the properties gained by the stochastic model also in terms of the continuous one. The probabilistic model checker PRISM permits the analysis of the transient behaviour of the stochastic model, e.g., the concentration of RafP at time t is given by:

$$\mathrm{C}_{RafP}(t) \;=\; \frac{0.1}{s} \cdot \underbrace{\sum_{i=1}^{4s} \big(i \cdot P(L_{RafP}(t) = i)\big)}_{expected\ value\ of\ L_{RafP}(t)} \;.$$

The random variable $L_{RafP}(t)$ stands for the level of RafP at time t. We set s to 1 for the 4 level version, and to 2 for the 8 level version. The factor $\frac{0.1}{s}$ calibrates the expected value for a given level to the concentration scale. In the 4 level version a single level represents 0.1 mMol and 0.05 mMol in the 8 level version. Figure 3 shows the simulation results for the species MEK and Ras-GTP in the time interval [0..100] according to the continuous and the stochastic models respectively. These results confirm that 4 levels are sufficiently adequate to approximate the continuous model, and that 8 levels are preferable if the computational expenses are acceptable.

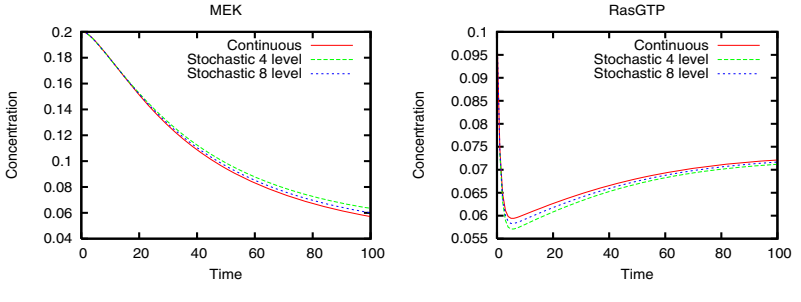


Fig. 3. Comparison of the concentration traces

Probabilistic model checking of special behavioural properties. We give two properties related to the partial order run of the I/O T-invariant, see Section 4.2 and qualitative property Q1 therein, from which we expect a consecutive increase of RafP, MEKPP and ERKPP. Both properties are expressed as so-called experiments, which are analysed varying the parameter L over all levels, i.e. 0 to N. For the sake of efficiency, we restrict the **U** operator to 100 time steps. Note that places are read as integer variables in the following.

Property S1: What is the probability of the concentration of RafP increasing, when starting in a state where the level is already at L (the latter side condition is specified by the filter given in braces)?

$$\mathbf{P}_{=?}\ [\ (\ \mathrm{RafP}\ =\ \mathrm{L}\ )\ \mathbf{U}^{<=100}\ (\ \mathrm{RafP} > \mathrm{L}\ )\ \{\ \mathrm{RafP} = \mathrm{L}\ \}\ ]$$

The results indicate, see Figure 4(a), that it is absolutely certain that the concentration of RafP increases from level 0 and likewise there is no increase from level N; this behaviour has already been determined by the qualitative analysis. Furthermore, an increase in RafP is very likely in the lower levels, increase and decrease are almost equally likely in the intermediate levels, while in the higher levels, but obviously not in the highest, an increase is rather unlikely (but not impossible). In summary this means that the total mass, circulating within the first layer of the signalling cascade, is unlikely to be accumulated in the activated form. We need this understanding to interpret the results for the next property.

Property S2: What is the probability that, given the initial concentrations of RafP, MEKPP and ERKPP being zero, the concentration of RafP rises above some level L while the concentrations of MEKPP and ERKPP remain at zero, i.e. RafP is the first species to react?

$$\mathbf{P}_{=?}\ [\ (\ (\ \mathrm{MEKPP}\ =\ 0\ )\ \wedge\ (\ \mathrm{ERKPP}\ =\ 0\ )\ )\ \mathbf{U}^{<=100}\ (\ \mathrm{RafP} >\ \mathrm{L}\ )$$
$$\{\ (\ \mathrm{MEKPP}\ =\ 0\ )\ \wedge\ (\ \mathrm{ERKPP}\ =\ 0\ )\ \ \wedge\ (\ \mathrm{RafP} =\ 0\ )\ \}\ ]$$

The results indicate, see Figure 4(b), that the likelihood of the concentration of RafP rising, while those of MEKPP and ERKPP are zero, is very high in the bottom half of the levels, and quite high in the lower levels of the upper half. The decrease of the likelihood in the higher levels is explained by property S1. Property S2 is related to the qualitative property Q1 (Section 4.2), and the

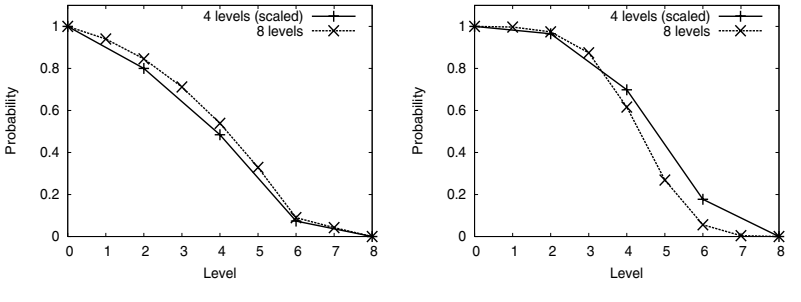


Fig. 4. Probability of the accumulation of RafP. (a) property S1. (b) property S2.

continuous property C1 (Section 6.2) – the concentration of RafP rises before those of MEKPP and ERKPP.

Due to the computational efforts of probabilistic model checking we are only able to treat properties over a stochastic model with 4 or at most 8 levels. This restricts the kind of properties that we can prove; e.g., in order to check increases of MEKPP and ERKPP – as suggested by the qualitative property Q1 and done above for RafP in the stochastic properties S1 and S2 – we would need 50 or 200 levels respectively.

Analytic probabilistic model checking becomes more and more impracticable with increasing size of the state space. Hence, the computation time of a probabilistic experiment, which typically consists of a series of probabilistic queries, can easily exceed several hours on a standard workstation. In order to avoid the enormous computational power required for larger state spaces, the time-dependent stochastic behaviour can be simulated by dedicated algorithms, e.g. [Gil77], or approximated by a continuous one, see next section.

6 The Continuous Approach

6.1 Continuous Modelling

In a continuous Petri net the marking of a place is no longer an integer, but a positive real number, which can be read as the concentration of the species modelled by the place. Transitions fire continuously, whereby the current deterministic firing rate generally depends on the current marking of the pre-places, i.e. of the current concentrations of the reactants. For our running case study, we derive the continuous model from the qualitative Petri net by associating a *mass action rate* with each transition in the network, i.e., the reaction labels are now read as the deterministic rate constants. We can likewise derive the continuous Petri net from the stochastic Petri net by approximating over the hazard function of type (1), see for instance [Wil06]. In both cases, we obtain a *continuous Petri net*, preserving the structure of the discrete one, see Figure 2.

The semantics of a continuous Petri net is defined by a system of ODEs, whereby one equation describes the continuous change over time on the token value of a given place by the continuous increase of its pre-transitions' flow and the continuous decrease of its post-transitions' flow, i.e., each place subject to changes gets its own equation. See [GH06] for more details.

The initial concentrations as suggested by the qualitative analysis correspond to those given in [LBS00], when mapping non-zero values to 1. For reasons of better comparability we have also considered more precise initial concentrations, where the presence of a species is encoded by biologically motivated real values varying between 0.1 and 0.4 in steps of 0.1. The complete system of non-linear ODEs generated from the continuous Petri net is given in [GHL07].

6.2 Continuous Analysis

Steady state analysis. Since there are 22 species, there are 2^{22} possible initial states in the qualitative Petri net (Boolean token interpretation). Of these,

118 were identified by the reachability graph analysis (Section 4.2) to form one strongly connected component, and thus to be "good" initial states. We then computed the steady state of the set of species for each possible initial state. In summary, our results show that all of the 'good' 118 states result in the same set of steady state values for the 22 species in the pathway, within the bounds of computational error of the ODE solver. None of the remaining possible initial states results in a steady state close to that generated by the 118 markings in the reachability graph; for details see [GHL07]. This is an interesting result, because the net considered here is not covered by the class of net structures discussed in [ADLS06] with the unique steady state property.

In Figure 5 (a) we reproduce the computed behaviour of MEK for all 118 good initial states, showing that despite differences in the concentrations at early time points, the steady state concentration is the same in all 118 states.

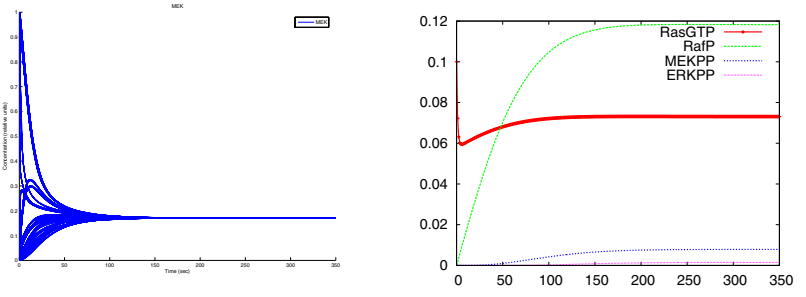


Fig. 5. (a) Steady state analysis of MEK for all 118 'good' states. (b) Continuous transient analysis of the phosphorylated species RasP, MEKPP, ERKPP, triggered by RasGTP.

Continuous model checking of the transient behaviour. Corresponding to the partial order run of the I/O T-invariant, see Section 4.2, we expect a consecutive increase of RafP, MEKPP, ERKPP, which we get confirmed by the transient behaviour analysis, compare Figure 5 (b). To formalise the visual evaluation of the diagram we use the continuous linear logic LTLc [CCRFS06], which is interpreted over the continuous simulation trace of ODEs.

The following three queries confirm together the claim of the expected propagation sequence. In the queries we have to refer to absolute values. The steady state values are obtained from the steady state analysis in the previous section; these are 0.12 mMol for RafP, 0.008 mMol for MEKPP and 0.002 mMol for ERKPP, all of them being zero in the initial state. If a species' concentration is above half of its steady state value, we call this concentration level significant. Note that in order to simplify the notation, places are interpreted as real variables in the following.

Property C1: The concentration of RafP rises to a significant level, while the concentrations of MEKPP and ERKPP remain close to zero; i.e. RafP is really the first species to react.

$$((\text{MEKPP} < 0.001) \wedge (\text{ERKPP} < 0.0002)) \ \mathbf{U} \ (\text{RafP} > 0.06)$$

Property C2: if the concentration of RafP is at a significant concentration level and that of ERKPP is close to zero, then both species remain in these states until the concentration of MEKPP becomes significant; i.e. MEKPP is the second species to react.

$$((\text{RafP} > 0.06) \wedge (\text{ERKPP} < 0.0002)) \Rightarrow ((\text{RafP} > 0.06) \wedge (\text{ERKPP} < 0.0002)) \mathbf{U} (\text{MEKPP} > 0.004)$$

Property C3: if the concentrations of RafP and MEKPP are significant, they remain so, until the concentration of ERKPP becomes significant; i.e. ERKPP is the third species to react.

$$((\text{RafP} > 0.06) \wedge (\text{MEKPP} > 0.004)) \Rightarrow ((\text{RafP} > 0.06) \wedge (\text{MEKPP} > 0.004)) \mathbf{U} (\text{ERKPP} > 0.0005)$$

Note that properties C1, C2 and C3 correspond to the qualitative property Q1, and that S2 is the stochastic counterpart of C1.

7 Tools

The bipartite graph in Figure 1 and its interpretation as the three Petri net models have been done using Snoopy [Sno], a tool to design and animate hierarchical graphs, including SBML import.

The qualitative analyses have been made using the Integrated Net Analyser INA [SR99] and the Model Checking Kit [SSE04]. We employed PRISM [PNK06] for probabilistic model checking, and Biocham [CCRFS06] for LTLc-based continuous model checking.

MATLAB [SR97] was used to produce the steady state analysis of all initial states in the continuous model, and the transient analysis was done using BioNessie [Bio], an SBML-based simulation and analysis tool for biochemical networks.

8 Summary and Outlook

In this paper we have described an overall framework that relates the three major ways of modelling biochemical networks – qualitative, stochastic and continuous – and illustrated this in the context of Petri nets. In doing so we have given a precise definition of biochemically interpreted stochastic Petri nets. We have shown that the qualitative time-free description is the most basic, with discrete values representing numbers of molecules or levels of concentrations. The qualitative description abstracts over two timed, quantitative models. In the stochastic description, discrete values for the amounts of species are retained, but a stochastic rate is associated with each reaction. The continuous model describes amounts of species using continuous values and associates a deterministic rate with each reaction. These two time-dependent models can be mutually approximated by hazard functions belonging to the stochastic world.

We have illustrated our framework by considering qualitative, stochastic and continuous Petri net descriptions of the ERK signalling pathway, based on the

model from Levchenko et al [LBS00]. We have focussed on analysis techniques available in each of these three paradigms, in order to illustrate their complementarity. Our special emphasis has been on model checking, which is especially useful for transient behaviour analysis, and we have demonstrated this by discussing related properties in the qualitative, stochastic and continuous paradigms. Although our framework is based on Petri nets, it can be applied more widely to other formalisms which are used to model and analyse biochemical networks.

We are now working on the incorporation of deterministic time into stochastic models, as well as the integration of continuous and stochastic aspects into one model.

Acknowledgements. The running case study has been partly carried out by Sebastian Lehrack during his study stay at the Bioinformatic Research Centre of the University of Glasgow. This stay was supported by the Max Gruenebaum Foundation [MGF] and the UK Department of Trade and Industry Beacon Bioscience Programme.

We would like to thank Richard Orton and Xu Gu for the constructive discussions as well as Vladislav Vyshemirsky for his support in the computational experiments.

References

[ADLS06] Angeli, D., De Leenheer, P., Sontag, E.D.: On the structural monotonicity of chemical reaction networks. In: ICATPN 2003, pp. 7–12. IEEE Computer Society Press, Los Alamitos (2006)

[Bio] BioNessie. A biochemical pathway simulation and analysis tool. University of Glasgow, http://www.bionessie.org

[BK02] Bause, F., Kritzinger, P.S.: Stochastic Petri Nets. Vieweg (2002)

[CCRFS06] Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 68–94. Springer, Heidelberg (2006)

[CGP01] Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press, Cambridge (2001)

[CKS07] Chickarmane, V., Kholodenko, B.N., Sauro, H.M.: Oscillatory dynamics arising from competitive inhibition and multisite phosphorylation. Journal of Theoretical Biology 244(1), 68–76 (2007)

[CVGO06] Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using continuous time Markov chains. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 44–67. Springer, Heidelberg (2006)

[DDS04] D'Aprile, D., Donatelli, S., Sproston, J.: CSL model checking for the GreatSPN tool. In: Aykanat, C., Dayar, T., Körpeoğlu, İ. (eds.) ISCIS 2004. LNCS, vol. 3280, pp. 543–552. Springer, Heidelberg (2004)

[GH06] Gilbert, D., Heiner, M.: From Petri nets to differential equations - an integrative approach for biochemical network analysis. In: Donatelli, S., Thiagarajan, P.S. (eds.) ICATPN 2006. LNCS, vol. 4024, pp. 181–200. Springer, Heidelberg (2006)

[GHL07] Gilbert, D., Heiner, M., Lehrack, S.: A unifying framework for modelling and analysing biochemical pathways using Petri nets. TR I-02, CS Dep., BTU Cottbus (2007)

[Gil77] Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry 81(25), 2340–2361 (1977)

[LBS00] Levchenko, A., Bruck, J., Sternberg, P.W.: Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. Proc. Natl. Acad. Sci. USA 97(11), 5818–5823 (2000)

[MGF] Max-Gruenebaum-Foundation, http://www.max-gruenebaum-stiftung.de

[Mur89] Murata, T.: Petri nets: Properties, analysis and applications. Proc.of the IEEE 77 4, 541–580 (1989)

[PNK06] Parker, D., Norman, G., Kwiatkowska, M.: PRISM 3.0.beta1 Users' Guide (2006)

[Sno] Snoopy. A tool to design and animate hierarchical graphs. BTU Cottbus, CS Dep., http://www-dssz.informatik.tu-cottbus.de

[SR97] Shampine, L.F., Reichelt, M.W.: The MATLAB ODE Suite. SIAM Journal on Scientific Computing 18, 1–22 (1997)

[SR99] Starke, P.H., Roch, S.: INA - The Intergrated Net Analyzer. Humboldt University, Berlin (1999), www.informatik.hu-berlin.de/~starke/ina.html

[SSE04] Schröter, C., Schwoon, S., Esparza, J.: The Model Checking Kit. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 463–472. Springer, Heidelberg (2003)

[Wil06] Wilkinson, D.J.: Stochastic Modelling for System Biology, 1st edn. CRC Press, New York (2006)

Appendix

The data files of the model in its three versions and the analysis results are available at www-dssz.informatik.tu-cottbus.de/examples/levchenko. A self-contained documentation of the case study as well as related work is given in [GHL07].

Reconstructing Metabolic Pathways by Bidirectional Chemical Search

Liliana Félix[1], Francesc Rosselló[2], and Gabriel Valiente[1]

[1] Algorithms, Bioinformatics, Complexity and Formal Methods Research Group, Technical University of Catalonia, E-08034 Barcelona, Spain
valiente@lsi.upc.edu
[2] Department of Mathematics and Computer Science, University of the Balearic Islands, E-07122 Palma de Mallorca
cesc.rossello@uib.es

Abstract. One of the main challenges in systems biology is the establishment of the metabolome: a catalogue of the metabolites and biochemical reactions present in a specific organism. Current knowledge of biochemical pathways as stored in public databases such as KEGG, is based on carefully curated genomic evidence for the presence of specific metabolites and enzymes that activate particular biochemical reactions. In this paper, we present an efficient method to build a substantial portion of the artificial chemistry defined by the metabolites and biochemical reactions in a given metabolic pathway, which is based on bidirectional chemical search. Computational results on the pathways stored in KEGG reveal novel biochemical pathways.

Keywords: Artificial chemistry, biochemical reaction, metabolic pathway.

1 Introduction

Metabolism can be regarded as a network of chemical reactions activated by enzymes and connected via their substrates and products, and a metabolic pathway can be regarded as a coordinated sequence of biochemical reactions [1]. The definition of a metabolic pathway is not exact, and most pathways constitute indeed highly intertwined cyclic networks. In a cell, the substrates of a pathway are usually the products of another pathway, and there are junctions where pathways meet or cross [2].

The analysis of metabolic pathways is motivated by the rapidly increasing quantity of available information on metabolic pathways for different organisms. One of the most comprehensive sources of metabolic pathway data is [3]. There are also several databases on metabolic pathways, such as aMAZE [4], BRENDA [5], MetaCyc [6], KEGG [7], and WIT [8]. These databases contain hundreds of metabolic pathways and thousands of biochemical reactions, and even the metabolic pathway for a small organism constitutes a large network. For instance, the proposed metabolic pathway for the bacterium *E. coli* consists

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 217–232, 2007.

of 436 compounds (substrates, products, and intermediate compounds) linked by 720 reactions [9].

An artificial chemistry [10], on the other hand, is a computational model of a chemical system that consists of a set of objects (molecules), a set of reaction rules (that allow for the production of new molecules from already existing molecules), and a definition of the dynamics of the system (that is, application conditions for the reaction rules), aimed at answering qualitative questions about the chemical system. Thus, artificial chemistries model real chemistries, in which molecules represent chemical compounds and reaction rules represent chemical reactions and, in particular, artificial chemistries model organic chemistries [11,12,13].

The chemical description of molecules in an artificial chemistry can be made at different levels of resolution, from simple molecular descriptors to structural formulas. One of these representations are *chemical graphs*, with nodes corresponding to the atoms of the molecules and edges indicating the bonds between them. Chemists have used chemical graphs to distinguish isomers since the second half of the nineteenth century. In first course Organic Chemistry classes, chemical reactions are explained in terms of constitutional formulas and a handful of reaction mechanisms, which are nothing but chemical graphs and rules to modify them by means of breaking, forming and changing the type of bonds. This leads in a natural way to artificial chemistries based on labeled graphs as molecules and graph transformation rules as reactions. Several such artificial chemistries have been proposed so far: see, for instance, [11,12,13,14,15].

Artificial chemistries can also be used to model biochemical systems such as metabolic pathways, in which molecules represent metabolites and reaction rules represent biochemical reactions [16], and they allow for answering qualitative questions about metabolism. In this paper, we present an efficient method to build a substantial portion of the artificial chemistry defined by the metabolites and biochemical reactions in a given metabolic pathway. Our method is based on bidirectional chemical search, and its implementation uses chemical graphs to represent sets of molecules. We report also on the results of some experiments applying this method to pathways stored in KEGG, which reveal novel biochemical pathways.

2 Modeling Biochemical Reactions as Chemical Graph Transformations

Following [15], by a *chemical graph* we understand a complete labeled weighted graph (V, E, ℓ, μ), with (V, E) an undirected graph (without multiple edges or self-loops), ℓ a labeling mapping that labels every node $v \in V$ with a chemical element $\ell(v)$, and $\mu : E \to \mathbb{N}$ an edge weight function. We shall denote the weight of the edge joining nodes v and w by $\mu(v, w)$; notice that $\mu(v, w) = \mu(w, v)$ because the graph is undirected. A weight of 0 stands for a non-existing bond, a weight of 1 for a single bond, a weight of 2 for a double bond, etc. The *valence* of a node in a chemical graph is the total weight of the edges incident to it.

To simplify the language, we shall call a *multimolecule* to any set of molecules. Such a multimolecule is described by the disjoint union of the chemical graphs

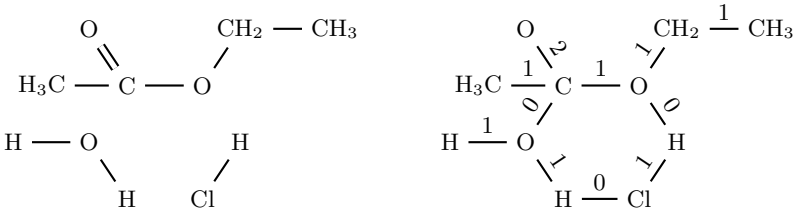


Fig. 1. A multimolecule and a simplified representation of it as a chemical graph. Only some weight 0 edges needed to make the graph connected are shown, for clarity.

representing the molecules and then adding weight 0 edges between atoms of different molecules. In this way, the molecules in the set are identified as maximal subgraphs with non-zero weight edges; see Fig. 1.

Given two chemical graphs $G_1 = (V_1, E_1, \ell_1, \mu_1)$ and $G_2 = (V_2, E_2, \ell_2, \mu_2)$, an *atom mapping* between them is a bijection $M : V_1 \to V_2$ such that, for every $v_1 \in V_1$:

- $\ell_1(v_1) = \ell_2(M(v_1))$
- $\sum_{w_1 \in V_1} \mu_1(v_1, w_1) = \sum_{w_1 \in V_1} \mu_2(M(v_1), M(w_1))$.

When there exists an atom mapping between two chemical graphs G_1 and G_2, these chemical graphs (and the multimolecules they represent) are said to be *compatible*: this means that they have the same number of nodes for each possible pair (label, valence).

A *chemical reaction graph* is a structure $R = (G_1, G_2, M)$, where $G_1 = (V_1, E_1, \ell_1, \mu_1)$ and $G_2 = (V_2, E_2, \ell_2, \mu_2)$ are compatible chemical graphs, called the *substrate* and the *product* chemical graphs respectively, and $M : V_1 \to V_2$ is an atom mapping between them.

The application of a chemical reaction graph to a given chemical graph, consists of breaking, forming and changing bonds in a subgraph of the chemical graph which is isomorphic to the substrate of the chemical reaction graph. Reversible chemical reaction graphs can also be applied in the opposite direction, by breaking, forming and changing bonds in a subgraph of the chemical graph which is isomorphic to the product of the chemical reaction graph.

The *size* of an atom mapping M between two chemical graphs $G_1 = (V_1, E_1, \ell_1, \mu_1)$ and $G_2 = (V_2, E_2, \ell_2, \mu_2)$ is given by

$$\text{size}(M) = \sum_{(v,w) \in E_1} |\mu_2(M(v), M(w)) - \mu_1(v, w)| \,.$$

Given two compatible chemical graphs $G_1 = (V_1, E_1, \ell_1, \mu_1)$ and $G_2 = (V_2, E_2, \ell_2 \mu_2)$, an *optimal* atom mapping between them is an atom mapping of minimal size, which always exists (but it needs not be unique). An optimal atom mapping models the classical principle of minimum structure change, by which a chemical reaction normally occurs through the redistribution of the minimum number of valence electrons, that is, the formation and breaking of the least number of covalent bonds [17].

The *size* of a chemical reaction graph $R = (G_1, G_2, M)$ is simply the size of the corresponding atom mapping M.

3 Reconstructing Metabolic Pathways by Bidirectional Chemical Search

Artificial chemistries [10] are computational models of chemical systems and, in particular, of biochemical systems such as metabolic pathways. An artificial chemistry consists of a set of *molecules*, a set of *reaction rules* that produce new molecules from already existing molecules, and the definition of the *dynamics* of the system, which specifies the application conditions of the rules, the preference in their application, etc. [16].

A metabolic pathway can be regarded as a coordinated sequence of biochemical reactions and is often described in symbolic terms, as a succession of transformations of one set of *substrate* molecules into another set of *product* molecules [18]. Substrate and product must be compatible chemical graphs for a pathway between them to exist [15,16,18].

Metabolic pathways are often represented as directed hypergraphs, with substrate and product molecules as nodes and biochemical reactions as hyperarcs. Since a chemical graph can represent the disjoint union of a set of molecules, though, the equivalent representation of artificial chemistries and, in particular, metabolic pathways as directed graphs becomes more natural. An artificial chemistry defined by a set of chemical reaction graphs, is thus represented as a directed second-order graph with the chemical graphs that represent the sets of substrate and product molecules as vertices and applications of the chemical reaction graphs, including information on atom mapping, as arcs.

Unfortunately, the size of the artificial chemistry defined by a set M of chemical graphs and a set R of chemical reaction graphs is often exponential in the size of M and R and thus, artificial chemistries are known for very small instances only, involving a few dozens of molecules and biochemical reactions. Therefore, we consider in this paper the problem of obtaining a substantial portion of the artificial chemistry defined by a set of biochemical reactions while avoiding the complexity of reconstructing the whole artificial chemistry.

The constraints we impose on the reconstruction process are threefold:

(1) The *initial chemical graphs* represent all sets of at most m metabolites among those involved in the set R or reactions, for some fixed, but arbitrary, m (in examples and applications in this paper we shall always take $m = 2$).
(2) The reconstruction process is restricted to a fixed, but arbitrary, number k of derivation steps.
(3) The initial and final sets of metabolites of every metabolic pathway belong to the set of initial chemical graphs.

While the first two constraints (on the size of the initial chemical graphs and the lengths of the metabolic pathways under inspection) are motivated by complexity considerations alone, the third constraint allows for directing the

search of new metabolic pathways *inside* the artificial chemistry. That is, instead of building the artificial chemistry by applying the biochemical reactions in every possible way to each of the initial chemical graphs, we perform a bidirectional search by constructing *forward* metabolic pathways of length at most k starting in initial chemical graphs and *backward* metabolic pathways of length at most k ending in initial chemical graphs, and then gluing them to obtain all metabolic pathways of length at most $2k$ *starting and ending* in initial chemical graphs.

Given a set R of biochemical reactions and a number k of derivation steps, the detailed procedure for reconstructing all metabolic pathways of length up to $2k$ using the metabolites and reactions in R and starting and ending in multimolecules of at most m components, is the following:

- First, we extract the set M of all chemical graphs representing sets of at most m any metabolites appearing in substrates and products of the reactions in R. We call the elements of M the *initial chemical graphs*.
- Next, we identify all compatibility classes in M (maximal subsets of compatible initial chemical graphs). Biochemical reactions transform chemical graphs into compatible chemical graphs, and therefore the origin and the end of a metabolic pathway will be compatible sets of metabolites. Thus, since we restrict ourselves to metabolic pathways starting and ending in initial chemical graphs, we can restrict ourselves to search for metabolic pathways starting and ending in each compatibility class of initial chemical graphs.
- Then, each compatibility class C in M is considered as potential substrates $C_F^{(0)}$ and potential products $C_R^{(0)}$ for the reactions in R.
- For every $i = 1, \ldots, k$, the forward application of the reactions in R to the elements of $C_F^{(i-1)}$ produces a set of multimolecules $C_F^{(i)}$, while the reverse application of these reactions to the molecules in $C_R^{(i-1)}$ produces a set of multimolecules $C_R^{(i)}$.
- Any nonempty intersection of a set obtained by forward application and a set obtained by reverse application of reactions yields a new pathway between elements of C. To avoid repetitions, it is enough to check whether each $C_F^{(i)}$ intersects $C_R^{(i)}$ and $C_R^{(i-1)}$. More specifically:
  - For $i = 1$, the forward application of the reactions in R to the molecules in $C_F^{(0)}$ produces a set $C_F^{(1)}$ of new molecules, and the reverse application of the reactions in R to the molecules in $C_R^{(0)}$ produces a set $C_R^{(1)}$ of new molecules.

$$C_F^{(0)} \rightarrow C_F^{(1)} = C_R^{(1)} \leftarrow C_R^{(0)}$$

    Then,
    - every member of $C_F^{(1)} \cap C_R^{(0)}$ yields a new pathway $C_F^{(0)} \rightarrow C_F^{(1)} \cap C_R^{(0)}$ of length 1;
    - every member of $C_F^{(1)} \cap C_R^{(1)}$ yields a new pathway $C_F^{(0)} \rightarrow C_F^{(1)} \cap C_R^{(1)} \rightarrow C_R^{(0)}$ of length 2.

- For $i = 2$, the forward application of the reactions in R to the molecules in $C_F^{(1)}$ produces a set $C_F^{(2)}$ of new molecules, and the reverse application of the reactions in R to the molecules in $C_R^{(1)}$ produces a set $C_R^{(2)}$ of new molecules.

$$C_F^{(0)} \to C_F^{(1)} \to C_F^{(2)} = C_R^{(2)} \leftarrow C_R^{(1)} \leftarrow C_R^{(0)}$$

  Then,
  - every member of $C_F^{(2)} \cap C_R^{(1)}$ yields a new pathway of length 3

$$C_F^{(0)} \to C_F^{(1)} \to C_F^{(2)} \cap C_R^{(1)} \to C_R^{(0)}.$$

  - every member of $C_F^{(2)} \cap C_R^{(2)}$ yields a new pathway of length 4

$$C_F^{(0)} \to C_F^{(1)} \to C_F^{(2)} \cap C_R^{(2)} \to C_R^{(1)} \to C_R^{(0)}.$$

- And, recursively, the forward application of the reactions in R to the molecules in $I_F = C_F^{(i-1)}$ produces a set $C_F = C_F^{(i)}$ of new molecules, and the reverse application of the reactions in R to the molecules in $I_R = C_R^{(i-1)}$ produces a set $C_R = C_R^{(i)}$ of new molecules.

$$C_F^{(0)} \to \cdots \to I_F \to C_F = C_R \leftarrow I_R \leftarrow \cdots \leftarrow C_R^{(0)}$$

  Then:
  - Every member of $C_F \cap I_R$ yields a new pathway of length $2i - 1$

$$C_F^{(0)} \to \cdots \to I_F \to C_F \cap I_R \to \cdots \to C_R^{(0)}.$$

  - Every member of $C_F \cap C_R$ yields a new pathway of length $2i$

$$C_F^{(0)} \to \cdots \to I_F \to C_F \cap C_R \to I_R \to \cdots \to C_R^{(0)}.$$

The following result shows that in this way we obtain all metabolic pathways of length at most $2k$ under constraints (1) and (3) above.

Lemma 1. *For every $i = 1, \ldots, k$, all metabolic pathways of length $2i - 1$ and $2i$ starting and ending in initial chemical graphs are obtained in the i-th iterative step of the procedure explained above.*

Proof. If

$$m_0 \to m_1 \to \cdots \to m_i \to \cdots \to m_{2i-1}$$

is a pathway with m_0 and m_{2i-1} initial chemical graphs, then $m_j \in C_F^{(j)}$ for every $j = 0, \ldots, i$ and $m_{2i-1-l} \in C_R^{(l)}$ for every $l = 0, \ldots, i - 1$, and hence, in particular, $m_i \in C_F^{(i)} \cap C_R^{(i-1)}$. Therefore, this path is obtained in the i-th iterative step of the procedure explained above.

On the other hand, if

$$m_0 \to m_1 \to \cdots \to m_i \to \cdots \to m_{2i}$$

is a pathway with m_0 and m_{2i} initial chemical graphs, then $m_j \in C_F^{(j)}$ for every $j = 0, \ldots, i$ and $m_{2i-l} \in C_R^{(l)}$ for every $l = 0, \ldots, i$, and hence, in particular, $m_i \in C_F^{(i)} \cap C_R^{(i)}$. Therefore, this path is also obtained in the i-th iterative step of that procedure. □

Example 1. Let a, b, c, d, e, f be metabolites such that b, d, e, f are compatible with each other, a is compatible with $b + b$ and c is compatible with $b + b + b$. Consider the toy artificial chemistry given by the following reactions (where only the first four reactions are reversible):

$$a + b \leftrightarrow c, \quad a \leftrightarrow d + e, \quad b + d \leftrightarrow b + e, \quad b + b \leftrightarrow d + f$$
$$c \to e + b + b, \quad d + d \to a, \quad a + f \to b + e + e$$

Let us look for metabolic pathways starting and ending with metabolites and pairs of metabolites $a, \ldots, e$ globally compatible with $b + b + b$. Then, the set M of all initial chemical graphs can be identified with the set of monomials of total weight at most 2 over the alphabet $\{a, b, c, d, e, f\}$ and the class C of the initial chemical graphs compatible with bbb (we omit henceforth the + sign for simplicity) is

$$C = \{c, ab, ad, ae, af\}.$$

So, we are looking for metabolic pathways starting and ending in elements of this set C. The intermediate multimolecules of these pathways will belong to the set of all multimolecules formed by metabolites a, b, c, d, e, f compatible with bbb: these are the multimolecules in C plus any combination of three metabolites b, d, e, f.

Taking

$$C_F^{(0)} = C_R^{(0)} = C = \{c, ab, ad, ae, af\},$$

we obtain the following one step derivations:

$C_F^{(0)}$	$\to$	$C_F^{(1)}$
c	$\to$	(ab, bbe)
ab	$\to$	(c, bde)
ad	$\to$	dde
ae	$\to$	dee
af	$\to$	(def, bee)

$C_R^{(1)}$	$\to$	$C_R^{(0)}$
(def, ddf)	$\to$	af
(dde, dee)	$\to$	ae
(dde, ddd)	$\to$	ad
(c, bde, bdd)	$\to$	ab
ab	$\to$	c

Notice that some elements of $C_F^{(1)}$ and $C_R^{(1)}$ do no longer belong to M, as we warned.

Then

$$C_F^{(1)} = \{c, ab, bbe, bde, bee, dde, dee, def\}$$
$$C_R^{(1)} = \{c, ab, bdd, bde, ddd, dde, ddf, dee, def\}$$

and hence

$$C_F^{(1)} \cap C_R^{(0)} = \{ab, c\}, \quad C_F^{(1)} \cap C_R^{(1)} = \{ab, c, bde, dde, dee, def\}.$$

From these intersections we deduce that all metabolic pathways of lengths 1 and 2 starting and ending in C are

$$c \to ab, \quad ab \to c, \quad c \to ab \to c, \quad ab \to c \to ab, \quad ab \to bde \to ab,$$
$$ad \to dde \to ad, \quad ad \to dde \to ae, \quad ae \to dee \to ae, \quad af \to def \to af.$$

For $k = 2$ we obtain:

$C_F^{(0)}$	$\to$	$C_F^{(1)}$	$\to$	$C_F^{(2)}$	$C_R^{(2)}$	$\to$	$C_R^{(1)}$	$\to$	$C_R^{(0)}$
c	$\to$	(ab, bbe)	$\to$	$((c, bde), (bbd, def))$	$((af, bbe), bbd)$	$\to$	(def, ddf)	$\to$	af
ab	$\to$	(c, bde)	$\to$	$((ab, bbe), (ab, bdd, bee))$	(ad, ae)	$\to$	(dde, dee)	$\to$	ae
ad	$\to$	dde	$\to$	(ad, ae)	$(ad, \emptyset)$	$\to$	(dde, ddd)	$\to$	ad
ae	$\to$	dee	$\to$	ae	$(ab, (ab, bdd, bee), bde)$	$\to$	(c, bde, bdd)	$\to$	ab
af	$\to$	(def, bee)	$\to$	$((af, bbe), bde)$	(c, bde, bdd)	$\to$	ab	$\to$	c

Then

$$C_F^{(2)} = \{c, ab, ad, ae, af, bbd, bbe, bdd, bde, bee, def\}$$
$$C_R^{(2)} = \{c, ab, ad, ae, af, bbd, bbe, bdd, bde, bee\}$$

and hence

$$C_F^{(2)} \cap C_R^{(1)} = \{c, ab, bdd, bde, def\},$$
$$C_F^{(2)} \cap C_R^{(2)} = \{c, ab, ad, ae, af, bbd, bbe, bdd, bde, bee\}.$$

From these intersections we deduce that all metabolic pathways of lengths 3 and 4 starting and ending in C are

$c \to ab \to c \to ab$
$ab \to c \to ab \to c$
$af \to bee \to bde \to ab$
$ab \to bde \to ab \to c$
$c \to bbe \to bbd \to ddf \to af$
$c \to ab \to bde \to bdd \to ab$
$ab \to c \to ab \to bde \to ab$
$ab \to bde \to ab \to c \to ab$
$ab \to bde \to bdd \to ab \to c$
$ab \to bde \to bee \to bde \to ab$
$ad \to dde \to ae \to dde \to ae$
$af \to def \to af \to def \to af$
$af \to bee \to bde \to ab \to c$
$c \to ab \to bde \to ab$
$ab \to bde \to bdd \to ab$
$c \to bbe \to def \to af$
$c \to ab \to c \to ab \to c$
$c \to ab \to bde \to ab \to c$
$ab \to c \to ab \to c \to ab$
$ab \to c \to bbe \to def \to af$
$ab \to bde \to ab \to bde \to ab$
$ab \to bde \to bdd \to bde \to ab$
$ad \to dde \to ad \to dde \to ad$
$ae \to dee \to ae \to dde \to ae$
$af \to def \to bbe \to def \to af$
$af \to bee \to bde \to bdd \to ab$

As it can be seen in the previous example, the raw application of the procedure explained above generates all metabolic pathways of length up to $2k$ starting and ending in sets of at most m metabolites used by the reactions in R, but most of these metabolic pathways will be redundant, for instance because they are cyclic, or because they do not contain any new multimolecule that has not appeared in shorter metabolic pathways. Therefore, several reconstruction problems may be addressed in this context. In this work we consider only two of them: first, to produce all metabolic pathways of length up to $2k$, and second, to produce all

shortest metabolic pathways of length up to $2k$, in both cases under restrictions (1) to (3) made explicit above.

We give our reconstruction algorithms in full pseudocode next. The first one formalizes the procedure explained above.

Algorithm 1. *Given a set R of biochemical reactions and a number k of derivation steps, obtain the set of all metabolic pathways of length up to $2k$ using the metabolites and reactions in R starting and ending in sets of at most m metabolites among those involved in the reactions in R.*

$M \leftarrow$ substrate and product metabolites of the reactions in R
$M \leftarrow \bigcup_{j=1}^{m} M^j$
$E \leftarrow M/_{\cong}$ where $m \cong m'$ if and only if m and m' are compatible
foreach $C \in E$ **do**
  $I_F \leftarrow I_R \leftarrow C$
  foreach $i \leftarrow 1$ ***to*** k **do**
    $N_F \leftarrow \emptyset$
    foreach $m \in I_F$ **do**
      foreach $r \in R$ **do**
        foreach $n \leftarrow$ forward application of r to m **do**
          $N_F \leftarrow N_F \cup \{n\}$
    $N_R \leftarrow \emptyset$
    foreach $m \in I_R$ **do**
      foreach $r \in R$ **do**
        foreach $n \leftarrow$ reverse application of r to m **do**
          $N_R \leftarrow N_R \cup \{n\}$
    output $C \rightarrow \cdots \rightarrow I_F \rightarrow N_F \cap I_R \rightarrow \cdots \rightarrow C$
    output $C \rightarrow \cdots \rightarrow I_F \rightarrow N_F \cap N_R \rightarrow I_R \rightarrow \cdots \rightarrow C$
    $I_F \leftarrow N_F$
    $I_R \leftarrow N_R$

The first three lines of this algorithm produce the different compatibility classes of initial chemical graphs. Then, for each compatibility class C and for each $i = 1, \ldots, k$:

- It receives the sets $I_F = C_F^{(i-1)}$ and $I_R = C_R^{(i-1)}$ of the results of all direct and reverse applications, respectively, of $i-1$ consecutive rules in R to multimolecules in C (when $i = 1$, $C_F^{(0)} = C$ and $C_R^{(0)} = C$) and it produces the sets $C_F^{(i)}$ and $C_R^{(i)}$ of the results of all direct and reverse applications, respectively, of rules in R to multimolecules in I_F and I_R, respectively. That is, the sets of the results of all direct and reverse applications, respectively, of i consecutive rules in R to multimolecules in C.
- The lines starting with *output* call a procedure that outputs the list of all metabolic pathways of lengths $2i-1$ and $2i$ obtained so far. When $i = 1$:

- the first *output* line gives all length 1 pathways $m \to m_f^{(1)}$, with $m \in C$,
- the second *output* line gives all length 2 pathways $m \to m_r^{(1)} \to m'$ with $m, m' \in C$,

And when $i > 1$

- the first *output* line gives all length $2i - 1$ pathways

$$m \to m_f^{(1)} \to \cdots \to m_f^{(i-1)} \to m_f^{(i)} = m_r^{(i-1)} \to m_r^{(i-2)} \to \cdots \to m_r^{(1)} \to m'$$

with $m, m' \in C$,

- the second *output* line gives all length $2i$ pathways

$$m \to m_f^{(1)} \to \cdots \to m_f^{(i-1)} \to m_f^{(i)} = m_r^{(i)} \to m_r^{(i-1)} \to \cdots \to m_r^{(1)} \to m'$$

with $m, m' \in C$.

The next algorithm produces a metabolic network containing all metabolic pathways up to a given length.

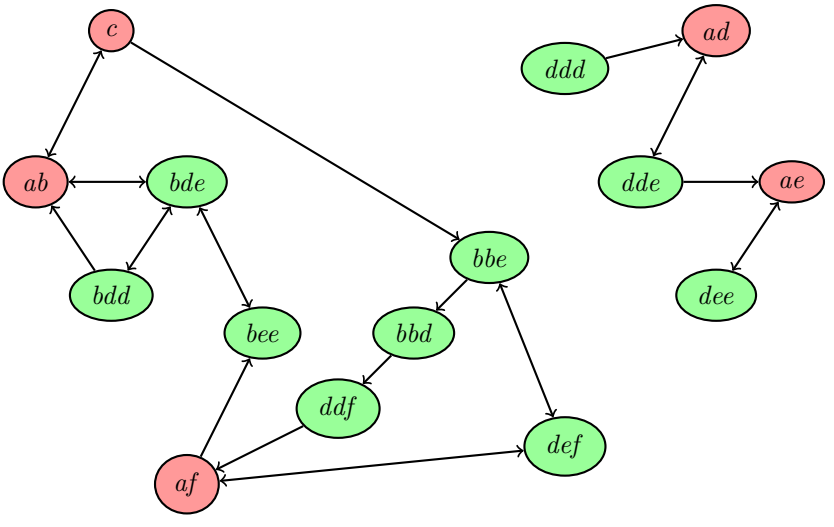
Algorithm 2. *Given a set R of biochemical reactions and a number k of derivation steps, obtain the metabolic network (X, Y) containing all metabolic pathways of length up to $2k$, using the metabolites and reactions in R starting and ending in sets of at most m metabolites among those involved in the reactions in R.*

```
M ← substrate and product metabolites of the reactions in R
M ← ⋃_{j=1}^{m} M^j
E ← M/≅ where m ≅ m' if and only if m and m' are compatible
X ← Y ← ∅
foreach C ∈ E do
    I_F ← I_R ← C
    foreach i ← 1 to k do
        N_F ← ∅
        foreach m ∈ I_F do
            foreach r ∈ R do
                foreach n ← forward application of r to m do
                    N_F ← N_F ∪ {n}
                    X ← X ∪ {m, n}
                    Y ← Y ∪ {(m, n)}
        N_R ← ∅
        foreach m ∈ I_R do
            foreach r ∈ R do
                foreach n ← reverse application of r to m do
                    N_R ← N_R ∪ {n}
                    X ← X ∪ {m, n}
                    Y ← Y ∪ {(n, m)}
        I_F ← N_F
        I_R ← N_R
return (X, Y)
```

Now, upon the metabolic network (X, Y) obtained with the previous algorithm, the set of all shortest metabolic pathways of length up to $2k$, using the metabolites and reactions in R starting and ending in sets of at most m metabolites among those involved in the reactions in R, can be obtained by using an all-pairs shortest path algorithm.

Example 2. The toy artificial chemistry of Example 1, obtained from the class $C = \{c, ab, ad, ae, af\}$ of the initial chemical graphs compatible with bbb by bidirectional search of metabolic pathways of length up to 4, is the following:

Then, the enumeration of all-pairs shortest paths in (X, Y) starting and ending in the elements of $C = \{c, ab, ad, ae, af\}$ produces the following derivations:



$c \to ab$
$c \to bbe \to def \to af$
$ab \to c$
$ab \to c \to bbe \to def \to af$
$ad \to dde \to ae$
$af \to bee \to bde \to ab$
$af \to bee \to bde \to ab \to c$

4 Results and Discussion

The metabolic reconstruction algorithm was implemented as a Perl script, using the `Chemistry::Reaction` module from the PerlMol collection of Perl modules for computational chemistry [19].

We have performed a series of experiments in order to reconstruct metabolic pathways for all known reference pathway maps. The protocol we have used is as follows.

Table 1. Number of new molecules (# mol) and new pathways (# path) of length up to $2k$ found by bidirectional chemical search upon the metabolites and reactions stored in KEGG for several reference maps, together with the number of compounds in the solved reactions and total number of compounds (# cpd) and the number of solved and total number of reactions (# rn) in each reference map

map	#	#	#	$k=1$		$k=2$		$k=3$		$k=4$		#	#
	cpd	mol	rn	mol	path	mol	path	mol	path	mol	path	mol	path
00020	20/38	230	10/29	569	1	552						1,351	1
00030	31/53	527	19/43	1,052	148	1,494	27	906		361		4,340	175
00120	31/62	527	18/46	2,148	2	1,102						3,777	2
00251	20/51	230	9/36	237	1	38		1				506	1
00260	61/91	1,952	32/66	934	2	406		288		67		3,647	2
00290	21/46	252	10/29	291	3	64						607	3
00340	37/60	740	19/38	1,350	1	504		1				2,595	1
00360	31/63	527	13/53	471	1	55						1,053	1
00410	34/56	629	16/34	1,370	1	1,566		2,717		2,058		8,340	1
00590	43/76	989	20/64	2,545	30	2,278		1,859		1,927		9,598	30
00906	38/100	779	23/103	685	6	167		12				1,643	6

1. Obtain reference pathway maps from the KEGG [20] database. We have used KEGG release 42.0 in all our experiments.
2. Solve the optimal atom mapping problem for all of the reactions in the reference pathways.
3. Reconstruct metabolic pathways of length up to 8 for each reference pathway.
4. Orient the reactions, according to the study of irreversibility of reactions in KEGG carried out in [21].
5. Filter out those metabolic pathways that involve irreversible reactions applied in the reverse direction.
6. Identify the new metabolites thus obtained, by chemical structure search in CheBi [22], MetaCyc [6], KEGG [20], and SciFinder Scholar [23].
7. Analyze the new metabolic pathways for coexistence of metabolites and enzymes in each particular organism.

Preliminary results obtained by following the aforementioned experimental protocol upon 11 of the 322 reference pathway maps in KEGG are summarized in Table 1. For the reference pathway map β-Alanine metabolism (00410), for instance, the optimal atom mapping problem was solved for 16 of the 34 reactions annotated to this pathway, which involve 34 of the 56 metabolites annotated to β-Alanine metabolism and thus, $34 + 34 \cdot 35/2 = 629$ initial and final molecules (metabolites and metabolite pairs). During the bidirectional chemical search for $k = 1$, the number of new metabolites was 1370 and one new metabolic pathway was obtained, and for $k = 2, 3, 4$, the number of new metabolites was 1566, 2717, and 2058, respectively, while no further new metabolic pathway was found and thus, one new metabolic pathway was found while generating 8340 new metabolites.

Among the novel metabolic pathways found by bidirectional search, using the metabolites and reactions stored in KEGG for carotenoid biosynthesis (reference pathway map 00906) we have obtained the following metabolic pathway:

```
C14146 + C13455 => R06958 => C14146 + C13456
  <= R06961 <= C08586 + C13456

alpha-Zeacarotene + Abscisic aldehyde
  => R06958 => alpha-Zeacarotene + Abscisic alcohol
  <= R06961 <= delta-Carotene + Abscisic alcohol
```

A KEGG pathway reference map contains information for several organisms. Thus, it is important to find evidence that all four metabolites appearing in this pathway are present in a same organism and also, that the enzyme activating the reverse biochemical reaction R06961 (carotene 7,8-desaturase, 1.14.99.30) is indeed expressed in that particular organism.

Carotenoid biosynthesis spans several related pathways: spheroidene, normal-spirilloxanthin, unusual-spirilloxanthin, abscisic acid biosynthesis, and astaxanthin biosynthesis. However, there are organisms whose metabolism does not include both carotenoid biosynthesis and abscisic acid biosynthesis. In fact, *Arabidopsis thaliana* (thale cress) is the only organism for which the four metabolites are annotated in KEGG to carotenoid biosynthesis, and the gene coding for carotene 7,8-desaturase, AT3G04870, is indeed expressed in *A. thaliana* [24,25].

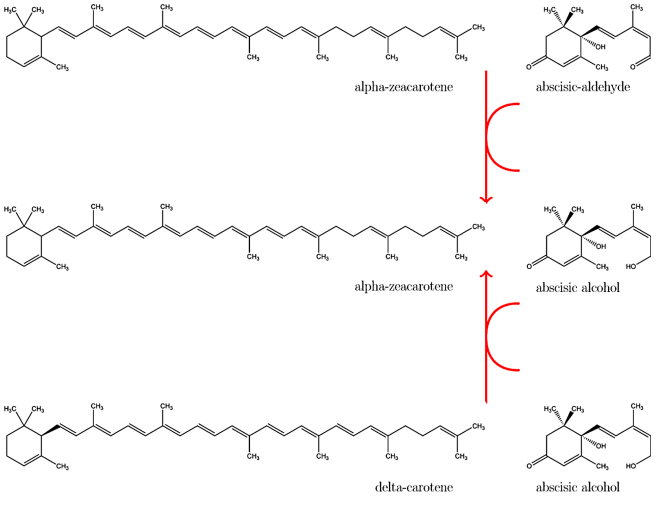


Fig. 2. A novel metabolic pathway found in the biosynthesis of steroids

On the other hand, there is a biosynthetic pathway, the plastidic 2C-methyl-D-erythritol 4-phosphate (MEP) pathway, that involves the four metabolites and occurs in plastids, protozoa, most bacteria, and algae [26]. In the MEP

Table 2. Number of potential biochemical reactions between sets of at most m metabolites among those involved in the reactions stored in KEGG for several reference maps. For each value of m, the first column gives the number of classes with two or more molecules (which indicates the possibility of a biochemical reaction among them) and the second column gives the total number of classes.

map	$m=1$		$m=2$		$m=3$		$m=4$		$m=5$	
00020	1	37	92	635	2,083	6,674	22,919	48,490	158,580	263,999
00030	7	41	253	688	3,511	6,580	27,476	41,628	144,858	192,953
00120	10	50	382	1,029	6,372	12,444	276,344	102,763	449,679	633,533
00251	3	48	199	1,082	5,145	14,886	69,825	136,383	568,520	870,618
00260	5	84	607	3,039	22,862	61,663	64,121	771,862	4,135,143	6,264,157
00290	8	37	260	618	3,646	6,258	30,895	43,888	183,141	231,422
00340	1	59	249	1,505	7,581	21,718	100,699	192,640	738,598	1,119,577
00360	7	51	354	1,070	5,957	11,920	52,707	83,189	297,431	406,549
00410	3	53	223	1,288	5,903	18,075	77,188	160,014	582,868	939,397
00590	10	29	207	385	2,081	3,168	13,660	18,541	66,703	84,116
00906	18	67	768	1,567	12,677	19,691	116,769	155,837	699,670	852,128

pathway, carotenoid biosynthesis is a precursor of abscisic acid biosynthesis [26, Fig. 1]. In the novel metabolic pathway, alpha-Zeacarotene (C14146) and delta-Carotene (C08586) are involved in carotenoid biosynthesis whereas Abscisic aldehyde (C13455) and Abscisic alcohol (C13456) are involved in abscisic acid biosynthesis. Such a possible link between the early and later stages of the biosynthesis of steroids was established in [26], where it is argued that only specific carotenoid intermediates (direct precursors of the abscisic acid biosynthesis) are increased or reduced, and further studied in [27] when regulating the early stages of abscisic acid biosynthesis in plants. The new metabolic pathway, shown in Fig. 2, is thus a novel pathway in the biosynthesis of carotenoid indeed.

While these preliminary results already reveal a number of new biochemical pathways, the artificial chemistry reconstruction starting from all sets of at most m metabolites among those involved in the set of reactions (the third constraint imposed on the reconstruction process) might reveal the existence of a much larger number of new biochemical pathways for $m > 2$. As can be seen in Table 2, the number of potential biochemical reactions grows fast with m for the reference maps stored in KEGG.

Another interesting avenue for further research is the extraction of all acyclic derivations from the artificial chemistry produced by Algorithm 2.

Acknowledgements

The research described in this paper has been partially supported by the Spanish CICYT, project TIN 2004-07925-C03-01 GRAMMARS and project MTM2006-07773 COMGRIO, and by EU project INTAS IT 04-77-7178.

References

1. Deville, Y., Gilbert, D., van Helden, J., Wodak, S.J.: An overview of data models for the analysis of biochemical pathways. Briefings in Bioinformatics 4, 246–259 (2003)
2. Karp, P.D., Mavrovouniotis, M.L.: Representing, analyzing, and synthesizing biochemical pathways. IEEE Expert 9, 11–21 (1994)
3. Michal, G.: Biological Pathways: An Atlas of Biochemistry and Molecular Biology. John Wiley & Sons, New York (1999)
4. Lemer, C., Antezana, E., Couche, F., Fays, F., Santolaria, X., Janky, R., Deville, Y., Richelle, J., Wodak, S.J.: The aMAZE LightBench: A web interface to a relational database of cellular processes. Nucleic Acids Research 32, 443–448 (2004)
5. Schomburg, I., Chang, A., Schomburg, D.: BRENDA, enzyme data and metabolic information. Nucleic Acids Research 30, 47–49 (2002)
6. Caspi, R., Foerster, H., Fulcher, C.A., Hopkinson, R., Ingraham, J., Kaipa, P., Krummenacker, M., Paley, S., Pick, J., Rhee, S.Y., Tissier, C., Zhang, P., Karp, P.D.: MetaCyc: A multiorganism database of metabolic pathways and enzymes. Nucleic Acids Res. 34, D511–D516 (2006)
7. Kanehisa, M., Goto, S.: KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Research 28, 27–30 (2000)
8. Overbeek, R., Larsen, N., Pusch, G.D., D'Souza, M., Selkov, E., Kyrpides, N., Fonstein, M., Maltsev, N., Selkov, E.: WIT: Integrated system for high-throughput genome sequence analysis and metabolic reconstruction. Nucleic Acids Research 28, 123–125 (2000)
9. Edwards, J.S., Palsson, B.O.: The Escherichia coli MG1655 in silico metabolic genotype: its definition, characteristics, and capabilities. Proc. Natural Academy of Science, USA 97, 5528–5533 (2000)
10. Dittrich, P., Ziegler, J., Banzhaff, W.: Artificial chemistries—a review. Artif. Life 7, 225–275 (2001)
11. Benkö, G., Flamm, C., Stadler, P.F.: A graph-based toy model of chemistry. J. Chem. Inf. Comput. Sci. 43, 1085–1093 (2003)
12. Benkö, G., Flamm, C., Stadler, P.F.: Generic properties of chemical networks: Artificial chemistry based on graph rewriting. In: Proc. 7th European Conf. Advances in Artificial Life. Lect. Notes Comput. Sci., vol. 2801, pp. 10–19. Springer, Heidelberg (2003)
13. Benkö, G., Flamm, C., Stadler, P.F.: Multi-phase artificial chemistry. In: Schaub, H., Detje, F., Brüggemann, U. (eds.) The Logic of Artificial Life: Abstracting and Synthesizing the Principles of Living Systems, pp. 16–22. IOS Press, Amsterdam (2004)
14. McCaskill, J., Niemann, U.: Graph replacement chemistry for DNA processing. In: Condon, A., Rozenberg, G. (eds.) DNA 2000. LNCS, vol. 2054, pp. 103–116. Springer, Heidelberg (2001)
15. Rosselló, F., Valiente, G.: Chemical graphs, chemical reaction graphs, and chemical graph transformation. Electr. Notes Theoret. Comput. Sci. 127, 157–166 (2005)
16. Rosselló, F., Valiente, G.: Graph transformation in molecular biology. In: Formal Methods in Software and System Modeling. Lect. Notes Comput. Sci., vol. 3393, pp. 116–133. Springer, Heidelberg (2005)
17. Temkin, O.N., Zeigarnik, A.V., Bonchev, D.: Chemical Reaction Networks: A Graph-theoretical Approach. CRC Press, Boca Raton, USA (1996)

18. Rosselló, F., Valiente, G.: Analysis of metabolic pathways by graph transformation. In: Proc. 2nd Int. Conf. Graph Transformation. Lect. Notes Comput. Sci., vol. 3256, pp. 70–82. Springer, Heidelberg (2004)
19. Tubert-Brohman, I.: Perl and chemistry. The Perl Journal 8, 3–5 (2004), PerlMol is available at: http://www.perlmol.org/
20. Kanehisa, M., Goto, S., Hattori, M., Aoki-Kinoshita, K.F., Itoh, M., Kawashima, S., Katayama, T., Araki, M., Hirakawa, M.: From genomics to chemical genomics: New developments in KEGG. Nucleic Acids Res. 34, D354–D357 (2006)
21. Ma, H., Zeng, A.-P.: Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms. Bioinformatics 2, 270–277 (2003)
22. Brooksbank, C., Cameron, G., Thornton, J.: The European Bioinformatics Institute's data resources: Towards systems biology. Nucleic Acids Res. 33, D46–D53 (2005)
23. Wagner, A.B.: Scifinder scholar 2006: An empirical analysis of research topic query processing. J. Chem. Inf. Model. 46, 767–774 (2006)
24. Bartley, G.E., Scolnik, P.A., Beyer, P.: Two *Arabidopsis thaliana* carotene desaturases, phytoene desaturase and ζ-carotene desaturase, expressed in *Escherichia coli*, catalyze a poly-*cis* pathway to yield pro-lycopene. Eur. J. Biochem. 259, 396–403 (1999)
25. Scolnik, P.A., Bartley, G.E.: Nucleotide sequence of Zeta-carotene Desaturase (accession no. U38550) from Arabidopsis. Plant Physiol. 109, 1499 (1995)
26. Estévez, J.M., Cantero, A., Reindl, A., Reichler, S., León, P.: 1-deoxy-D-xylulose-5-phosphate synthase, a limiting enzyme for plastidic isoprenoid biosynthesis in plants. J. Biol. Chem. 276, 22901–22909 (2001)
27. Seo, M., Koshiba, T.: Complex regulation of ABA biosynthesis in plants. Trends Plant Sci. 7, 41–48 (2002)

Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks

Aurélien Naldi, Denis Thieffry, and Claudine Chaouiya

TAGC, INSERM ERM206, Université de la Méditerranée
Marseilles, France

Abstract. The complexity of biological regulatory networks calls for the development of proper mathematical methods to model their structures and to obtain insight in their dynamical behaviours. One qualitative approach consists in modelling regulatory networks in terms of logical equations (using either Boolean or multi-valued discretisation).

In this paper, we propose a novel implementation of the generalised logical formalism by means of Multi-valued Decision Diagrams. We show that the use of this representation enables the development of efficient algorithms for the analysis of specific dynamical properties of the regulatory graphs. In particular, we address the question of determining conditions insuring the functionality of feedback circuits, as well as the identification of stable states. Finally, we apply these algorithms to logical models of T cell activation and differentiation.

Keywords: Regulatory networks, logical modelling, decision diagrams, regulatory circuits, stable states.

1 Introduction

Modelling is a crucial step towards a functional understanding of the complex interaction networks that govern fundamental cellular processes. In this respect, in order to overcome the lack of quantitative data, logical approaches have been successfully applied to a wide variety of genetic networks involved in cell differentiation and pattern formation (for an introduction to logical modelling of genetic networks, see [12,3]). However, when facing very large regulatory networks, even logical abstraction leads to hard combinatorial problems. In other contexts, decision diagrams have been successfully applied to similar combinatorial problems, in particular for symbolic model-checking (*e.g.* [2]). Here, we show how decision diagrams can be used to represent sophisticated logical rules and enable the development of efficient algorithms to determine the conditions insuring specific dynamical roles for the different regulatory circuits, as well as to identify all the stable states of large and complex systems, without explicitly constructing the state transition graph and independently of any initial conditions.

Finally, we apply these algorithms to: (i) a model of Th1/2 differentiation [8] and (ii) a model of the TCR signalling pathway [7].

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 233–247, 2007.

2 Logical Modelling of Gene Regulatory Networks

Our modelling approach leans on the generalised logical formalism initially developed by R. Thomas and collaborators [13,3]. In this context, a regulatory network and its dynamics are both represented in terms of oriented graphs.

2.1 Regulatory Graphs

A regulatory network is defined as a labeled directed graph $\mathcal{R} = (\mathcal{G}, \mathcal{A}, \mathcal{K})$ called *regulatory graph*, where:

- $\mathcal{G} = \{g_1, \ldots, g_n\}$ is the set of nodes of the regulatory graph, representing genes (or, more generally, regulatory components). Each $g_i \in \mathcal{G}$ is associated with its maximum *expression level* Max_i ($Max_i \in \mathbb{N}^*$) and its current expression level x_i ($x_i \in [0, Max_i]$).
- $\mathcal{A}$ is the set of arcs. An arc (g_i, g_j) specifies that the gene g_i regulates the gene g_j (when there is no possible confusion, we often write i for g_i). A regulatory graph may contain self-loops (*e.g.* a self-regulated gene g_i with an arc (i, i)).

  For each gene g_j, $Reg(j)$ denotes the set of its regulators: $i \in Reg(j)$ if and only if $(i, j) \in \mathcal{A}$.

  If $Max_i > 1$, g_i may have different effects onto a gene g_j, depending on the actual activity level of g_i. Thus the arc (i, j) may be indeed a multi-arc encompassing different *interactions*. The multiplicity of the arc (i, j) (*i.e.* the number of its constitutive interactions), is denoted $m_{i,j}$ ($1 \leq m_{i,j} \leq Max_i$). A *threshold* $\theta_{i,j,k}$ (integer taking its values in $[1, m_{i,j}]$) is associated to the k^{th} interaction (denoted $(i, j, k), k \in [1, m_{i,j}]$), with $1 \leq \theta_{i,j,1} < \ldots < \theta_{i,j,m_{i,j}} \leq Max_i$. The k^{th} interaction is *active*, when the level of its source g_i lays between the threshold of this interaction and that of the next interaction: $\theta_{i,j,k} \leq x_i < \theta_{i,j,k+1}$ (by convention, $\theta_{i,j,m_{i,j}+1} = Max_i + 1$).
- $\mathcal{K} = (\mathcal{K}_1, \ldots, \mathcal{K}_n)$ defines the dynamics of the system: each $\mathcal{K}_i$ is a multi-valued logical function defining the evolution of the variable x_i, depending on the incoming active interactions of g_i:

$$\mathcal{K}_i : \left(\prod_{j \in Reg(i)} [0, m_{j,i}] \right) \rightarrow [0, Max_i] \,.$$

  For example, if g_2 and g_3 are regulators of g_1 ($Reg(1) = \{2, 3\}$), $\mathcal{K}_1(0, 1)$ is the focal value of g_1 when no interaction from g_2 is active (*i.e.* $x_2 < \theta_{2,1,1}$) and the first interaction from g_3 is active ($\theta_{3,1,1} \leq x_3 < \theta_{3,1,2}$).

Note that the biologists often consider *different types* of interactions: activations (*resp.* repressions) have a positive (*resp.* negative) effect on their targets. However the actual effect of an interaction often depends on the presence of co-factors; its sign may even change depending on the context. In any case, the signs of interactions can be derived from the logical functions.

2.2 Dynamics of a Regulatory Graph

A state x of the regulatory graph is a n-tuple $(x_1, \ldots, x_n)$ of the expression levels of the genes: $x \in \prod_{i=0}^{n}[0, Max_i]$. Given a state and for each gene g_i, it is then possible to determine the set of interactions operating onto g_i (the active interactions). We thus define the functions $\mathcal{K}'_i(x)$s, which follow from the logical functions $\mathcal{K}_i$s and directly depend on the current state x of the system:

$$\begin{aligned} \mathcal{K}'_i : \prod_{j=1}^{n}[0, Max_j] &\longrightarrow [0, Max_i] \\ x &\longrightarrow \mathcal{K}_i\left(\sum_{k=1}^{m_{j,i}} k.\mathbf{1}_{[\theta_{j,i,k},\theta_{j,i,k+1}[}(x_j)\right)_{j\in Reg(i)} . \end{aligned}$$

where $\mathbf{1}$ denotes the indicator function.

In the following, for simplicity, $\mathcal{K}'_i$ will be denoted $\mathcal{K}_i$ (omitting the prime sign).

Given the current state x, the level of each g_i tends toward the *focal value* given by $\mathcal{K}_i(x)$. If this is greater (*resp.* lower) than x_i (the current value of g_i), there is a call to increase (*resp.* decrease) by one the value of g_i.

The dynamics of the regulatory graph can then be represented by a *state transition graph*, where nodes represent states (giving the levels of the regulatory components) and arcs represent transitions between states (*i.e.* changes of the values of some components). This state transition graph is computed by means of the $\mathcal{K}_i$s, which indicate the transitions leading from the current state to its following states (here, we consider an asynchronous updating, where each transition corresponds to a change of a unique variable, see [3] for further details). When facing large regulatory networks (*i.e.* dozens or hundreds of components), combinatorial problems impede the full computation of state transition graphs. In this paper, we assess the use of Decision Diagrams to handle this combinatorial problem for complex multi-valued logical models.

3 Regulatory Graphs and Multi-valued Decision Diagrams

Multi-valued logical functions have a finite number of possible values, depending on a set of *decision variables*, which also take a finite range of values. Such functions can be represented using efficient data structures (see [1] for further details).

Decision Diagrams are particularly promising in our context. Indeed, Garg et al. have already used BDD to represent the whole state transition graph of Boolean models of biological regulatory networks (their approach is contrasted with ours in the conclusion).

In the following section, we recall the definitions at the basis of our novel implementation of logical functions.

3.1 Decision Diagrams

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be represented as a *binary decision tree* where non-terminal nodes are labelled by a decision variable and where terminal nodes are labelled either 1 (true) or 0 (false). The edge from a decision node to its left child (*resp.* right child) corresponds to an assignment of the variable to 0 (*resp.* to 1). Given a state $x \in \{0,1\}^n$ (defining the values of n decision variables), a unique path from the root to a terminal node (a leaf) is defined. Along this path, the child chosen for each non-terminal node is labelled with the value of the corresponding variable in state x. The terminal node reached through this path gives the value of $f(x)$.

Reduced Binary Decision Diagrams (RBDDs) have been introduced to improve this representation, which requires exponential space ($2^{n+1} - 1$ nodes). RBDDs are obtained applying two reduction rules: (i) merge isomorphic subgraphs and (ii) bypass nodes whose children are the roots of isomorphic subdiagrams. The resulting structure is then a rooted directed acyclic graph. Bryant further extended this representation by the use of a fixed variable ordering which leads to canonical representations of logical functions. The resulting graphs are called Reduced *Ordered* BDDs (ROBDDs), commonly referred to as BDDs. Note that the size of BDDs may depend on the variable ordering (see Figure 1 for an illustration of the impact of the ordering).

BDDs have been generalised to the multi-valued case: a discrete multi-valued function can be represented by a *Multi-valued Decision Diagram* (MDD), where decision nodes may have as many children as the number of their possible values and the terminal nodes are labelled by the values of the function (see Figure 1 for an illustration) [6]. The ordering and reduction rules defined for BDDs apply also to this multi-valued generalisation.

3.2 Use of MDDs to Represent Logical Functions

The functions $\mathcal{K}_i$s, which take their values in $[0, Max_i]$, can be represented as MDDs, with the regulators of g_i as decision variables. During the simulation (*i.e.* the construction of the state transition graph), the focal value of a gene g_i is obtained in $O(\#Reg(i))$ in the worst case, traversing the corresponding MDD. This data structure thus definitely improved the performance of GINsim, our software implementing the logical formalism [5].

Moreover the representation of the $\mathcal{K}_i$s by means of MDDs greatly facilitates the analysis of specific dynamical properties as showed in the following sections.

In the sequel, for simplicity, diagrams will be named after the multi-valued functions they represent. For a given regulatory graph, an arbitrary ordering on the set of variables is used consistently for all the related MDDs.

4 Analysis of Regulatory Circuits

In what follows, we consider *elementary circuits* (circuits, for short), *i.e.* finite closed paths in the regulatory graph, where all the nodes are distinct.

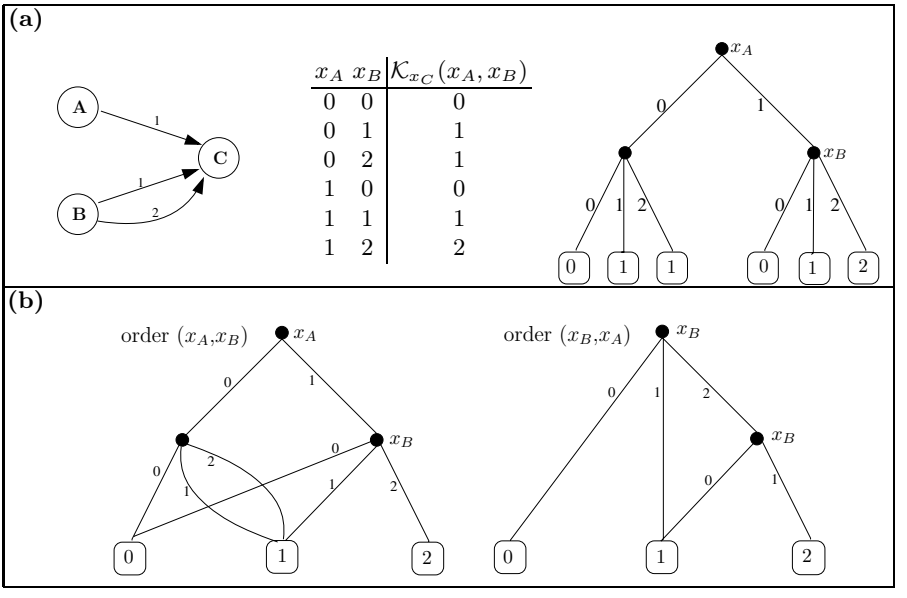
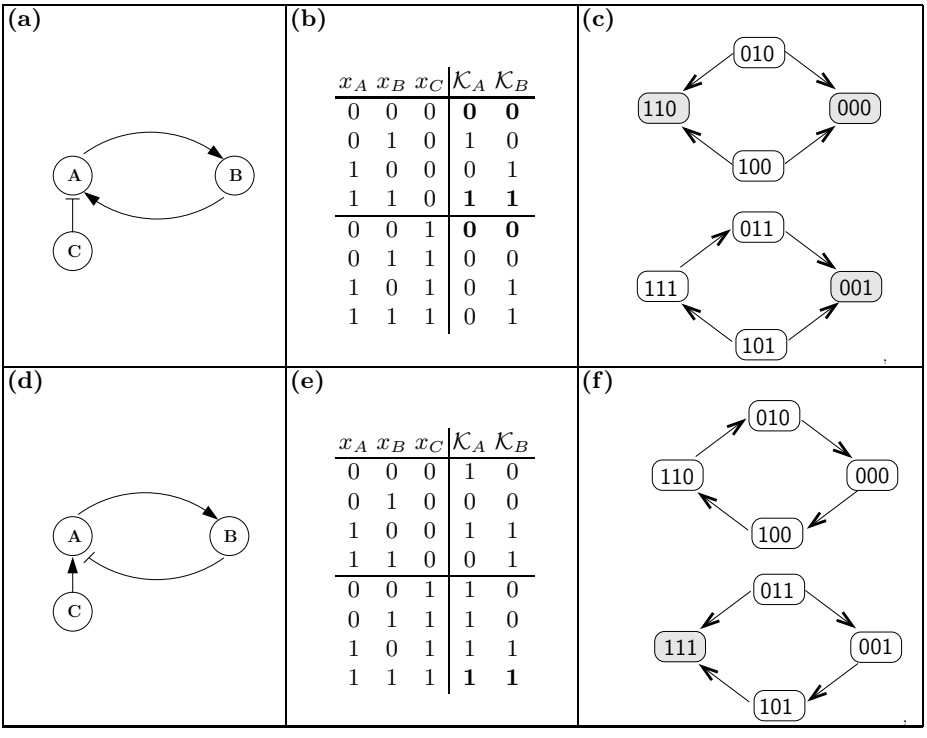


Fig. 1. Example of a simple logical regulatory graph with its MDD representation: **(a)** The regulatory graph, the table defining the function $\mathcal{K}_C$ together with its decision tree representation. **(b)** The reduced MDDs (considering the two different ordering of x_A and x_B) representing $\mathcal{K}_C$, with x_A and x_B as decision variables and the values of $\mathcal{K}_C$ labelling the leaves.

It is well established that complex dynamical behaviours of regulatory networks are related to their topological structures. In particular, the roles of regulatory circuits have been emphasised by R. Thomas, who proposed that negative circuits are required to observe oscillatory behaviours, whereas positive circuits are necessary for multistationarity (the sign of a circuit is given by the product of the signs of its interactions) [11]. Proofs of these conditions have been presented within different modelling formalisms [10,9]. But the sole presence of a circuit in a network does not guarantee the appearance of the corresponding dynamical behaviour. The circuit must be *functional* [11]. Figure 2 illustrates how the regulators of one of its components can prevent a circuit from generating the expected behaviour. We thus define the *context of functionality* of a circuit as a set of constraints on the values of the external inputs acting on that circuit. Our definition of functionality context may serve as a basis to formally prove the relationship between the functionality of a circuit and the corresponding dynamical properties.

In the multi-valued case, circuits containing multiple interactions can be splitted into multiple "elementary circuits" and considered separately. In the sequel we restrict ourselves to the case of single interactions to simplify the notation: the threshold of an interaction can be thus denoted $\theta_{i,j}$ (instead of $\theta_{i,j,k}$).



x_A	x_B	x_C	$\mathcal{K}_A$	$\mathcal{K}_B$
0	0	0	**0**	**0**
0	1	0	1	0
1	0	0	0	1
1	1	0	**1**	**1**
0	0	1	**0**	**0**
0	1	1	0	0
1	0	1	0	1
1	1	1	0	1

x_A	x_B	x_C	$\mathcal{K}_A$	$\mathcal{K}_B$
0	0	0	1	0
0	1	0	0	0
1	0	0	1	1
1	1	0	0	1
0	0	1	1	0
0	1	1	1	0
1	0	1	1	1
1	1	1	**1**	**1**

Fig. 2. Illustration of the influence of external inputs on the dynamics of a regulatory circuit. In the regulatory graphs of panels **(a)** and **(d)**, activations are depicted by normal arrows whereas inhibitions are depicted by blunt arrows. **(a)** A simple positive circuit affected by a negative input. **(b)** Values for $\mathcal{K}_A$ and $\mathcal{K}_B$: two situations arise, depending on the value of x_C (component A regulated by both B and C). **(c)** The two possible behaviours, depending on x_C: for $x_C = 0$ there are two stable states (top), while a unique stable state exists for $x_C = 1$ (bottom). **(d,e,f)** A simple negative circuit affected by a positive input; oscillations only appear in the absence of the input.

4.1 Functionality Context and Sign of an Interaction

In general, we say that an interaction (i, j) is functional when it affects the focal value of its target: $\mathcal{K}_j(x_1, \ldots, \theta_{i,j} - 1, \ldots, x_n) \neq \mathcal{K}_j(x_1, \ldots, \theta_{i,j}, \ldots, x_n)$. In the context of a circuit, this change must further affect the activity of the next interaction in the circuit (the threshold of the next interaction must be crossed to reach the focal value). This additional constraint is only relevant for multi-valued genes, as it is always satisfied in the Boolean case.

In the following, we define the functionality of the interaction (i, j) and its context, that is the set of constraints upon the regulators of j (target of the considered interaction).

Definition 1. *Let consider an interaction (i, j), member of a circuit $\mathcal{C}$ with a threshold $\theta_{i,j}$. Let (j, k) be the next interaction in $\mathcal{C}$, with a threshold $\theta_{j,k}$. The*

interaction (i,j) is said to be functional in $\mathcal{C}$ if and only if there exists a variable assignment for all regulators of g_j except g_i such that:

$$\mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j}-1,\ldots,x_n) < \theta_{j,k} \leq \mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j},\ldots,x_n), \quad (1)$$

$$\text{or} \quad \mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j},\ldots,x_n) < \theta_{j,k} \leq \mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j}-1,\ldots,x_n). \quad (2)$$

The functionality context of the interaction (i,j) in $\mathcal{C}$ is defined as the subset of $\Pi_{k=1}^{n}[0,Max_k]$ of n-tuples such that the values of the regulators of g_j let the interaction (i,j) functional (i.e. Equation (1) or (2) satisfied). The interaction is thus functional if its context is not empty.

Definition 1 establishes that the interaction (i,j) is functional provided its activity affects the activity of the following interaction of the circuit (going out g_j). This depends on the values of $\mathcal{K}_j$, considering values $\theta_{i,j}-1$ and $\theta_{i,j}$ for g_i and all possible values of other regulators of g_j.

We can then define the sign of an interaction, as 0 when it is not functional, or 1 (*resp.* -1) when it is functional and leads to an increase (*resp.* a decrease) of the focal value of its target.

Definition 2. *Let us consider consecutive interactions (i,j) and (j,k) in a circuit $\mathcal{C}$. Given a variable assignment for all regulators of g_j (except g_i), the sign of (i,j) in $\mathcal{C}$ is given by $\Gamma_{i,j}$, defined as follows:*

$$\Gamma_{i,j}(x) = \begin{cases} 1 & \text{if Equation (1) holds,} \\ -1 & \text{if Equation (2) holds,} \\ 0 & \text{otherwise,} \end{cases}$$

where x is a state ($x \in \Pi_{k=1}^{n}[0,Max_k]$), for which the values corresponding to the regulators of g_j (except that of g_i) equal the given assignment.

We say that (i,j) has a positive effect when $\Gamma_{i,j}(x)=1$, a negative effect when $\Gamma_{i,j}(x)=-1$ and no effect otherwise.

The construction of the MDD representing $\Gamma_{i,j}$ for a given interaction (i,j) is illustrated in Figure 3**(a-b)**. The algorithm is given as supplementary material. The leaves of the MDD give the sign of (i,j), depending on the path followed to reach them. This path defines the conditions on the values taken by the regulators of g_j.

Remark 1. It may happen that the sign of an interaction (i,j) is *context dependent*, that is, for an assignment of the regulators of g_j (except g_i) the sign of the interaction is positive and for another assignment, it is negative. To simplify the explanations in the next section (that defines the functionality of a whole circuit), we exclude such a case, which is infrequent in genetic regulatory networks.

4.2 Functionality Context and Sign of a Regulatory Circuit

We now consider the case of a whole elementary circuit. Definition 3 formalises the functionality context of a circuit, as well as its sign, depending on its constitutive interactions.

First, we will consider the case of circuits which do not contain smaller circuit(s), or shortcuts. A circuit $\mathcal{C} = (c_1, c_2, \ldots, c_r)$ (with $r + 1 = 1$) contains a shortcut if there exists c_i which regulates c_k, with $k \neq i + 1$ ($c_i \in \mathcal{C}$ and $c_k \in \mathcal{C}$). The simplest example of such a shortcut is an auto-regulation of a member of the circuit. Then, we extend the definition of the functionality to the general case (with, for simplicity, the restriction that no interaction of the circuit has a context dependent sign).

Definition 3. *The functionality context of a circuit with no shortcut is defined as the intersection of the functionality contexts of its constitutive interactions; the circuit is thus functional when this intersection is not empty (implying that all its interactions are functional). The sign of the circuit $\mathcal{C} = (c_1, c_2, \ldots, c_r)$ is defined as the product of the signs of its interactions:*

$$\Gamma_{\mathcal{C}}(x) = \prod_{i=1}^{i \leq r} \Gamma_{c_i, c_{i+1}}(x).$$

Definition 4. *Let consider a circuit $\mathcal{C} = (c_1, c_2, \ldots, c_r)$ which contains shortcut(s). Its functionality context $\Gamma_{\mathcal{C}} \subseteq \Pi_{k=1}^{n}[0, Max_k]$ is defined as the intersection of the contexts of its interactions further restricted to insure that, for all c_i acting on a component c_k of $\mathcal{C}$ different from c_{i+1}, and an assignment $\bar{y}$ of all variables but c_i:*

$$(y_1, \ldots, \theta_{i,i+1} - 1, \ldots y_n) \in \Gamma_{\mathcal{C}} \Longleftrightarrow (y_1, \ldots, \theta_{i,i+1}, \ldots y_n) \in \Gamma_{\mathcal{C}}.$$

When the above condition does not hold, all the tuples $(y_1, \ldots, x_i, \ldots y_n)$ are removed from $\Gamma_{\mathcal{C}}$ (for all possible values x_i of c_i). The circuit $\mathcal{C}$ is functional if its functionality context $\Gamma_{\mathcal{C}}$ is not empty. Its sign is defined as the product of the signs of its interactions.

The above definition guarantees that, given a fixed assignment of all other variables compatible with $\Gamma_{\mathcal{C}}$ (the functionality context of $\mathcal{C}$), both states where c_i level is less or equal to $\theta_{i,i+1}$ (the threshold of the interaction of $\mathcal{C}$ going from c_i to c_{i+1}) are in $\Gamma_{\mathcal{C}}$ (and this insures the functionality of this interaction).

The determination of the function Γ_C requires two operations: (i) the determination of all the $\Gamma_{i,j}$, giving the signs of single interactions; and (ii) the computation of their product.

Assume that the diagrams giving the signs of the interactions composing the circuit have been determined. The MDD giving the global sign of the circuit is built as a product of the signs of the individual interactions. This product is performed by means of the combination of MDDs Γ_1 and Γ_2, applying the following rules (see Figure 3**(c)** for an illustration):

- if $\Gamma_1$and Γ_2 are reduced to single nodes, the product is a node, its value being the product of those of $\Gamma_1$and Γ_2;
- else if Γ_1 (resp Γ_2) is a single node with value 1, the result is Γ_2 (*resp.* Γ_1);

- else if Γ_1 and Γ_2 roots are internal nodes with the same order (hence corresponding to the same decision variable), the result is a root of this order, and its children are recursive combinations of those of Γ_1 and Γ_2 roots;
- otherwise, if Γ_2 is a single node with value (-1) or if the root of Γ_1 has an order less than that of the root of Γ_2 (or symmetrically), the result is a root such that:
  - its order is the order of the root of Γ_1;
  - its children are recursive combinations of Γ_2 with those of Γ_1.

Finally, in the case of shortcuts in the circuit, as the sign of the circuit $\mathcal{C}$ may depend on the levels of members of $\mathcal{C}$, this dependancy is properly removed while ensuring that every member of the circuit can cross its threshold. Further details can be found in the supplementary material.

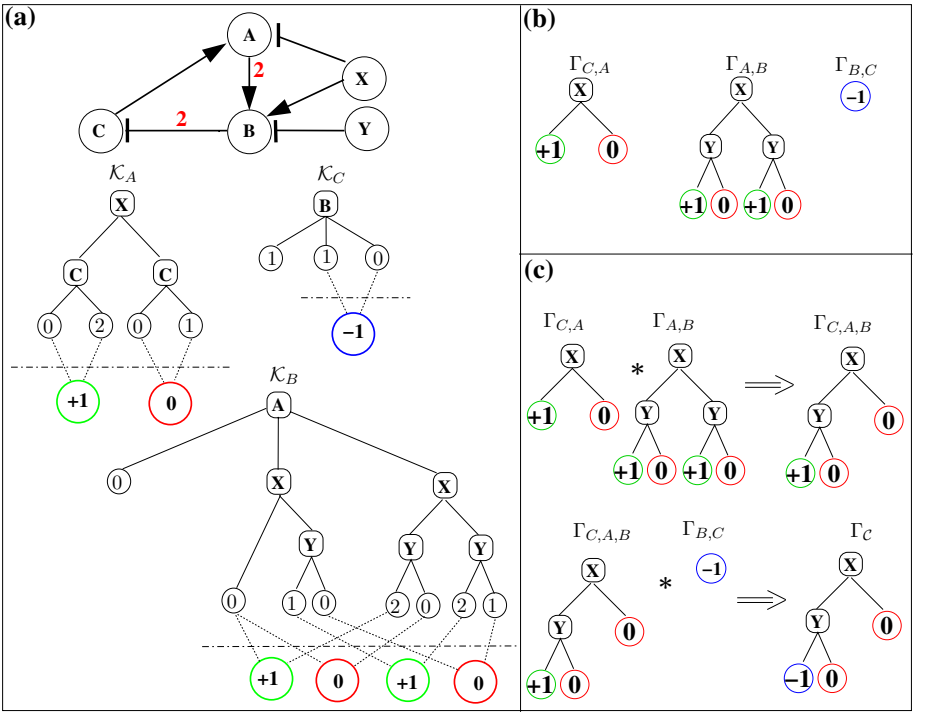


Fig. 3. Determination of the functionality context (and sign) of a circuit: **(a)** A regulatory network encompassing a circuit and decision diagrams of the logical functions $\mathcal{K}$s for the three members of the circuit. Below the logical functions $\mathcal{K}$s, pairwise comparisons of the leaves delineate the signs of the interactions targeting the members of the circuit. **(b)** MDDs giving the signs Γs of the interactions. **(c)** Combinations of the MDDs to determine $\Gamma_{\mathcal{C}}$, the sign of the circuit. The paths in $\Gamma_{\mathcal{C}}$ leading to non-zero leaves give the functionality context of the circuit. For sake of clarity, the diagrams are not fully reduced.

5 Efficient Determination of Logical Stable States

In this section, we show how the MDD representation of the logical functions $\mathcal{K}_i$s can be used to determine all the logical stable states of a parameterised regulatory graph. A stable state x is such that the focal value of each gene is identical to its current value:

$$x \in \Pi_{i=1}^{n}[0, Max_i] \text{ is stable } \textit{iff } \mathcal{K}_i(x) = x_i, \ \forall i \in \{1, \ldots n\}. \tag{3}$$

The algorithm to determine the stable states encompasses two main steps. First, for each gene g_i, a MDD $\mathcal{S}_i$ is constructed, which gives the logical stability condition depending on its value x_i and on those of its regulators (box (b) in Figure 4). Second, the resulting MDDs are combined as described in 4.2.

For a gene g_i, the first step amounts to transform the MDD representing the logical function $\mathcal{K}_i$. The decision variable x_i is properly added and the leaves values are set to 0 for a change (a decrease or an increase), or to 1 for no change. The resulting MDD implements a logical function with value 1 (true) when the node is stable, 0 (false) otherwise:

$$\mathcal{S}_i : \prod_{j \in Reg(i)} [0, Max_j] \rightarrow \{0, 1\},$$

with

$$\mathcal{S}_i = \begin{cases} 0 \text{ if } \mathcal{K}_i(x) \neq x_i, \\ 1 \text{ if } \mathcal{K}_i(x) = x_i. \end{cases}$$

To simplify the notation, $S_i(x)$ and $K_i(x)$ are considered beyond the sole regulators of i, to encompass all components of x. The diagrams $\mathcal{S}_i$ are then pairwise combined (the combination of $\mathcal{S}_i$ and $\mathcal{S}_j$ is the representation of the logical stability condition for both g_i and g_j). As for the determination of the sign of a circuit, each combination amounts to the product of the corresponding MDDs. Ultimately, the diagram $\mathcal{S}_{1\ldots n}$ defines the stability condition for the whole set of genes, hence the paths leading to 1-leaves give the stable states (see Figure 4).

6 Application to Regulatory Networks Controlling T Cell Activation and Differentiation

We have applied our novel analysis tools to two logical models recently published: (i) the first (multi-valued) model accounts for the differentiation of naive T-helper lymphocytes into two subtypes, called Th1 and Th2, controlling cellular and humoral immune responses, respectively [8]; (ii) the second, a Boolean model, integrates the information available on T cell receptor (TCR) signalling, taking into account several co-acting signals [7]. These two models are closely related, as NFAT, one of the outputs of the TCR signalling pathway, is one of the activators of the IFN-γ pathway.

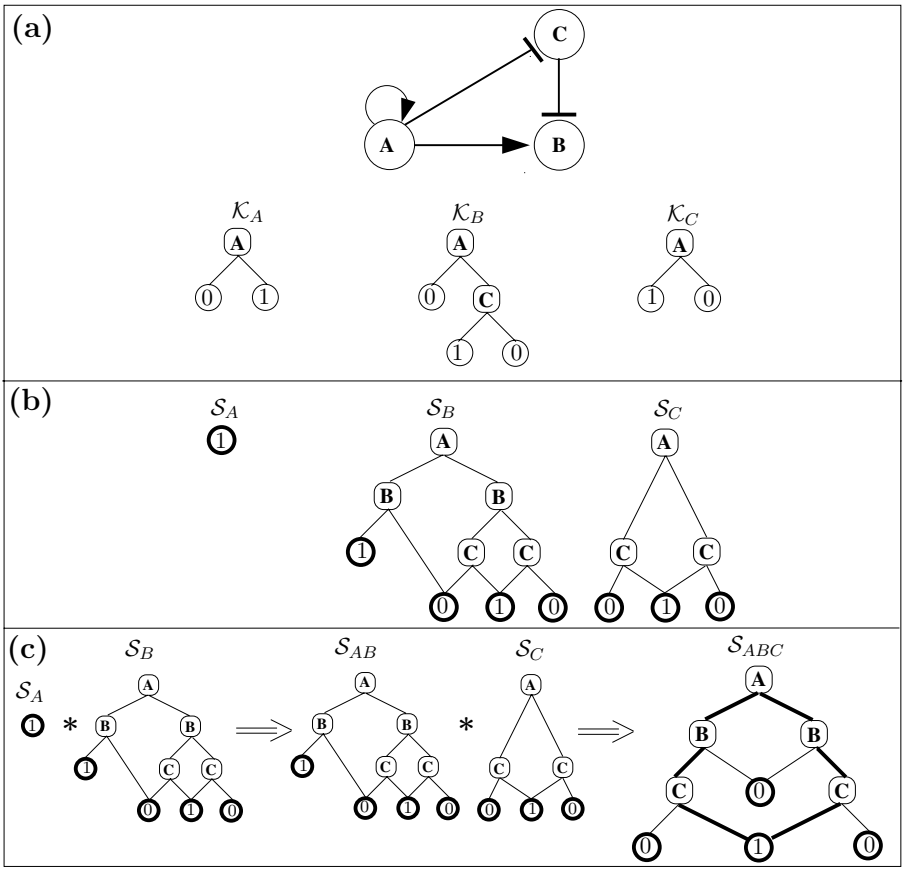


Fig. 4. Illustration of the stable state determination: **(a)** a simple regulatory graph and the associated logical functions; **(b)** the diagrams $\mathcal{S}$ giving the constraints for each gene to be stable; **(c)** the combination of the diagrams to obtain $\mathcal{S}_{ABC}$. For $\mathcal{S}_{ABC}$, two paths lead to the 1-leaf (they are depicted in bold), indicating that the regulatory graph has two stable states: $(x_A, x_B, x_C) = (0, 0, 1)$ and $(1, 1, 0)$. Note that, for clearness, the diagrams are not fully reduced.

6.1 T Cell Differentiation

The regulatory graph is shown in Figure 5. The graph encompasses 17 regulatory components, including five cytokines or intercellular signalling molecules, the interferons beta and gamma, and the interleukines 4, 12 and 18, the corresponding receptors, five mediatory molecules, SOCS1, IRAK, and STAT1, 4 and 6, as well as two transcription factors, Tbet and GATA3. All components but four are modelled by Boolean variables. The cascade involving IFN-γ, its receptor, STAT1 and Tbet are modelled by ternary variables as cells presenting two different levels of activation of the IFN-γ pathway have been

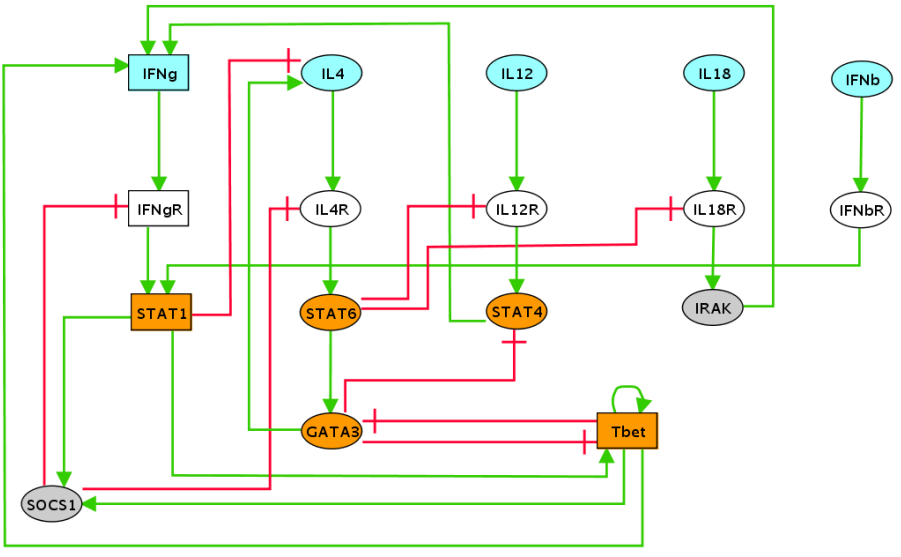


Fig. 5. The network controlling Th1/2 differentiation. All regulatory components but four are modelled by Boolean variables. The remaining four (rectangular nodes) are modelled by ternary variables. The first two nodes layers denote cytokines and their receptors. Normal arrows represent activations, whereas blunt arrows represent inhibitory effects. Note that the original model of Mendoza encompasses two variants, including an auto-activation for GATA3 (murine cells) or not (human cells).

experimentally observed. The experimental information in support of this graph, as well as the delineation of the logical parameters can be found in the original article published by Mendoza [8]. The logical model can be downloaded in a GINsim dedicated XML format from the address http://gin.univ-mrs.fr/GINsim/.

The algorithm presented in Section 5 takes less than a second to identify the four stable states reported by Mendoza:

- a *Th0* state without any active component, corresponding to the naive, undifferentiated cell;
- a *Th1* state where IFN-γ, IFN-γR, STAT1, SOCS1 and Tbet are expressed;
- a *Th1** state, similar to the previous one, but with higher expression levels for IFN-γ, and T-bet (the expression of SOCS1 prevents IFN-γR and STAT1 from showing a higher expression level);
- a *Th2* state showing expression of IL4, IL4R, STAT6 and GATA3.

Turning to the circuit functionality analysis, the algorithm described in Section 4 leads to the identification of five functional circuits among the 22 circuits found in the regulatory graph for the human model variant shown in Figure 5. All of these functional circuits are positive, which is consistent with the fact that this network is predominantly involved in the control of cell differentiation.

- The auto-regulation of T-bet is functional in the absence of both GATA3 and STAT1, or yet in the presence of a medium level of STAT1. Note that this circuit involves two different levels of Tbet and can thus enable the presence of up to three stable states, each characterized by one of the possible expression levels of Tbet.
- The (GATA3, IL4, IL4R, STAT6) circuit is functional in the absence of STAT1, SOCS1 and T-bet. This circuit ensures a coupling between the expressions of GATA3, IL4, IL4R and STAT6. In the murine cells, this circuit is not functional, replaced by the GATA3 auto-activation (functional in the absence of Tbet and STAT6).
- The (GATA3, T-bet) circuit is functional in the presence of STAT1 and STAT6. This cross-inhibitory circuit ensures the exclusive expressions of the transcriptional regulators Tbet and GATA3, characteristic of Th1 and Th2 responses, respectively. In the absence of the activators of Tbet and GATA3 (STAT1 and STAT6), the cell remains trapped in the naive state.
- The last two functional circuits involve IFN-γ, IFN-γR, STAT1, SOCS1, STAT6, and STAT4, plus a few additional components. Their contexts of functionality are relatively restrictive (absence of Tbet and IFNbR, and presence of IL12 and IL4, plus the absence of GATA3 in one of the two cases).

On the basis of the results of the circuit functionality analysis, it is possible to delineate specific perturbations affecting the stable state configuration.

6.2 T Cell Activation

The model recently published by Klamt et al. for T cell activation encompasses 40 regulatory components, hierarchically organised, from cell membrane receptors (TCR, CD8 and CD45) to transcription factors (CRE, AP1, NF-κB, NFAT) [7]. After adding three functional auto-activations on the input nodes (TCR, CD8 and CD45), our stable state identification tool identifies seven alternative stable states, each corresponding to a specific input configuration. Strikingly, the input configuration with all receptors permanently activated does not lead to any stable state, but rather to a complex periodic behaviour. However, in normal physiological situations, one should expect only transient activations of these receptors, thus ultimately leading to a unique stable state, corresponding to the resting situation.

In this respect, cross-talks between the signalling cascades may play an important role. Here, our circuit functionality analysis tool can be useful. Apart from the auto-activations purposively added on each of the three receptors, its application to this system leads to the identification of only one negative functional circuit out of nine: (ZAP70, cCBL). Under full stimulation (*i.e.* in the presence of all three inputs), this circuit enables an oscillatory behaviour of the involved regulatory components. These oscillations propagate downstream, leading to cyclic expression of the transcription factors (outputs of this model). In physiological situations, these oscillations must abort following the natural receptor inactivation.

7 Conclusion

In this paper, we have shown how Multi-valued Decision Diagrams (MDDs) can be used to encode the logical functions governing the behaviours of individual nodes in qualitative models of complex regulatory networks. Leaning on this encoding, we have further delineated two algorithms, one to efficiently determine all stable states of a logical model, the other to compute the functionality context of each regulatory circuit, in terms of conditions on the values of the variables acting on this circuit.

As mentionned above, Garg *et al.* have already represented Boolean state transition graphs in terms of BDD. They considered the particular case of networks where genes are expressed provided all their inhibitors are absent and at least one of their activators are present [4]. Based on their BDD representation, the authors propose an efficient method to determine stable states, and even more complex attractors. In contrast, our proposal refer to multi-valued logical networks where the logical functions can be more subtle. More importantly, our approach is based on a Decision Diagram representation of the transition functions for each node. Stable states, as well as feedback circuit functionality, are then determined through proper combinations of these diagrams.

On the basis of a prototype implementation of these algorithms, we are presently analysing a series of regulatory models (Boolean or multi-valued) involved in cell differentiation and pattern formation, encompassing dozens of regulatory components, involved in hundreds of regulatory circuits. For each of these systems, the delineation of all stable states and the computation of feedback circuit functionality domains took less than a second on a standard computer.

In section 6, we have briefly presented the results obtained for two recently published logical models: (i) a multi-valued model of the network controlling T-helper lymphocytes differentiation; (ii) a Boolean model encompassing some of the main signalling cascades controlling the activation of T cells.

At this point, we believe that it should be possible to further improve the performance of our algorithms. In particular, a proper ordering of decision variables can have a significant impact on the overall sizes of the MDDs (note that a common ordering of the variables must be defined to ensure a coherent combination of the MDDs). Similarly, we observed that the order of consecutive MDD combinations (*e.g.* in the course of the identification of all stable states) has a strong effect on the overall performance.

Finally, this MDD representation opens interesting prospects for the modelling of combinations of mutations or other perturbations through a rewriting of the MDDs describing the wild-type model.

Supplementary Material

Further details on the algorithms, as well as on the T-helper cell differentiation and activation models are available at the following URL:
http://gin.univ-mrs.fr/GINsim/publications/naldi2007.html.

Acknowledgements

We acknowledge financial support from the European Commission (contract LSHG-CT-2004-512143), the French Research Ministry through the ANR project JC05-53969 and A. Naldi PhD grant.

References

1. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Comput. 35, 677–691 (1986)
2. Burch, J.R., Clarke, E.M., Long, D.E., MacMillan, K.L., Dill, D.L.: Symbolic Model Checking for Sequential Circuit Verification. IEEE Trans. Comput.-Aided Design Integrated Circuits 13, 401–424 (1994)
3. Chaouiya, C., Remy, E., Mossé, B., Thieffry, D.: Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. Lect. Notes Comp. Inf. Sci. 294, 119–126 (2003)
4. Garg, A., Xenarios, I., Mendoza, L., DeMicheli, G.: An Efficient Method for Dynamic Analysis of Gene Regulatory Networks and in-silico Gene Perturbation Experiments. Lect. Notes Comp. Sci. 4453, 62–76 (2007)
5. González, A., Naldi, A., Sanchez, L., Thieffry, D., Chaouiya, C.: GINsim: A software suite for the qualitative modelling, simulation and analysis of regulatory networks. Biosystems 84, 91–100 (2006)
6. Kam, T., Villa, T., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: Multi-valued decision diagrams: Theory and applications. Int. J. Multiple-Valued Logic 4, 9–12 (1998)
7. Klamt, S., Saez-Rodriguez, J., Lindquist, J.A., Simeoni, L., Gilles, E.D.: A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinformatics 7(56) (2006)
8. Mendoza, L.: A network model for the control of the differentiation process in Th cells. Biosystems 84, 101–114 (2006)
9. Remy, E., Ruet, P., Mendoza, L., Thieffry, D., Chaouiya, C.: From logical regulatory graphs to standard petri nets: Dynamical roles and functionality of feedback circuits. Lect. Notes Comp. Sci. 4230, 55–72 (2006)
10. Soulé, C: Graphic requirements for multistationarity. ComPlexUs 1, 123–133 (2003)
11. Thomas, R.: On the relation between the logical structure of systems and their ability to generate multiple steady states of sustained oscillations. Springer Series Synergetics 9, 180–193 (1988)
12. Thomas, R.: Regulatory networks seen as asynchronous automata: A logical description. J. Theor. Biol. 153, 1–23 (1991)
13. Thomas, R., Thieffry, D., Kaufman, M.: Dynamical behaviour of biological regulatory networks–I. biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull. Math. Biol. 57, 247–276 (1995)

Author Index