Monika Heiner Hiroshi Matsuno (Eds.)

BioPPN 2011

International Workshop on Biological Processes & Petri Nets

Workshop co-located with PETRI NETS 2011
Newcastle upon Tyne, United Kingdom, June 20, 2010
Proceedings

Editors' addresses:
Monika Heiner
Brandenburg University of Technology at Cottbus
Faculty of Mathematics, Natural Sciences and Computer Science
Postbox 10 13 44
D-03013 Cottbus, Germany
monika.heiner@informatik.tu-cottbus.de

Hiroshi Matsuno
Yamaguchi University
Graduate School of Science and Engineering
1677-1 Yoshida, Yamaguchi
753-8512 Japan
matsuno@sci.yamaguchi-u.ac.jp

Preface

These proceedings contain the nine peer-reviewed contributions accepted for the Second International Workshop on Biological Processes & Petri Nets (BioPPN 2011), held as a satellite event of PETRI NETS 2011, in Newcastle, UK, on June 20, 2011. This workshop has been organised as a communication platform for researchers interested in the application of Petri nets in the broad field of *integrative biology*.

Integrative biology aims at deciphering essential biological processes that are driven by complex mechanisms, involving miscellaneous interacting molecular compounds. In this context, the need for appropriate mathematical and computational modelling tools is widely advocated. Petri nets have proved their usefulness for the modelling, analysis, and simulation of a diversity of biological networks, covering qualitative, stochastic, continuous and hybrid models. The deployment of Petri nets to study biological applications has not only generated original models, but has also motivated fundamental research.

We received two types of contributions: research papers and work-in-progress papers. All papers have been reviewed by four to six reviewers coming from or being recommended by the workshop's Program Committee. The list of reviewers comprises 30 professionals of the field. The nine accepted papers (with an acceptance rate of 82%) involve 41 authors coming from 9 different countries. In summary, the workshop proceedings enclose theoretical contributions as well as biological applications, demonstrating the interdisciplinary nature of the topic.

The workshop was complemented by an invited talk *Systems Biology in Supercomputing Environment* given by Satoru Miyano from Human Genome Center, Institute of Medical Science at the University of Tokyo in Japan.

As the hosting conference was originally planned to take place in Japan, it was an explicit goal of our workshop to promote the communication between Europe and Asia. We are especially glad that three papers from Japanese authors made it into the workshop. We would like to express our sympathy with the Japanese people and do hope that the precautions by the Steering committee will turn out to be overprotecting ones.

For more details see the workshop's website
http://www-dssz.informatik.tu-cottbus.de/BME/BioPPN2011.

June 2011 Monika Heiner and Hiroshi Matsuno

Program Committee

David Angeli, I
Gianfranco Balbo, I
Rainer Breitling, NL, UK
Jorge Carneiro, PT
Claudine Chaouiya, F, PT
Austin H. Chen, TW
Ming Chen, CN
David Gilbert, UK
Simon Hardy, CA
Ralf Hofestädt, D
Hsueh-Fen Juan, TW
Peter Kemper, US
Hanna Klaudel, F
Dong-Yup Lee, KR, SG
Chen Li, CN, JP
Wolfgang Marwan, D
Eduardo Mendoza, Philippines, D
Hirotada Mori, JP
Morikazu Nakamura, JP
P.S. Thiagarajan, SG
Toshimitsu Ushio, JP

Subreviewers

Liu Bing
Francesca Cordero
Lukasz Fronc
Blaise Genest
Andras Horvath
Sylvain Sen
Jeremy Sproston

Contents

Systems Biology in Supercomputing Environment

Satoru Miyano

Human Genome Center, Institute of Medical Science, The University of Tokyo
4-6-1 Shirokanedai, Minatoku, Tokyo 108-8639, Japan

We developed a software platform XiP (eXtensible integrative Pipeline) that is a flexible, editable and modular environment with a user-friendly interface for systems biology. With XiP, we can build various analysis workflows including simulation, visualization, pathway modeling, etc. Construction of a workflow under XiP follows the intuitive notion of dragging and dropping. XiP already equips with several "ready to use" pipeline flows for the most common analysis and more than 300 statistical/computational analysis components, and XiP also recognizes components written in R (http://www.r-project.org). It runs under multiprocessors environments. The source code is available as open source under the Lesser General Public License (LGPL) (download: http://xip.hgc.jp/wiki/en/).

On the other hand, we have been developing a modeling and simulation tool Cell Illustrator Online 5.0 (CIO, https://cionline.hgc.jp/) that enables us to draw, model, elucidate and simulate complex biological processes and systems such as metabolic pathways, signal transduction cascades, gene regulatory pathways and dynamic interactions of various biological entities. Cell System Ontology (CSO) (http://www.csml.org/) and Cell System Markup Language (CSML) consitute the basis of CIO. CIO uses Hybrid Functional Petri Net with extension (HFPNe) which is defined by enhancing some functions to hybrid Petri net so that various aspects in biological pathways can be intuitively modeled. CIO has been used to build, for example, a simulatable macrophage pathway knowledge base (MACPAK) (http://macpak.csml.org/). A key technology for driving systems biology is a method for automatic parameter estimation for models. For this purpose, we developed a method for HFPNe, namely CIO, by using a technolgy called data assimilation which "blends" simulation models and observational data "rationally". This data assimilation method requires supercomputer systems.

We are currently creating the next generation supercomputing environment for systems biology on XiP by enrolling CIO, data assimilation, model checking, and various systems biology tools such as SiGN (large-scale gene network estimation software package (http://sign.hgc.jp/) on XiP.

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.7-20

Optimal Control of Asynchronous Boolean Networks Modeled by Petri Nets

Koichi Kobayashi and Kunihiko Hiraishi

Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan
Tel:+81-761-51-1282, Fax:+81-761-51-1149
{k-kobaya,hira}@jaist.ac.jp

Abstract. A Boolean network model is one of the models of gene regulatory networks, and is widely used in analysis and control. Although a Boolean network is a class of discrete-time nonlinear systems and expresses the synchronous behavior, it is important to consider the asynchronous behavior. In this paper, using a Petri net, a new modeling method of asynchronous Boolean networks with control inputs is proposed. Furthermore, the optimal control problem of Petri nets expressing asynchronous Boolean networks is formulated, and a solution method is proposed. The proposed approach provides us a new control method of gene regulatory networks.

1 Introduction

In recent years, there have been a lot of studies on modeling, analysis, and control of gene regulatory networks in both the control community and the theoretical biology community. Gene regulatory networks are in general expressed by ordinary/partial differential equations with high nonlinearity and high dimensionality. In order to deal with such a system, it is important to consider a simple model, and various models such as Bayesian networks, Boolean networks, hybrid systems (piecewise affine models), and Petri nets have been developed so far (see e.g., [14]). In control problems, Boolean networks and hybrid systems are frequently used [1, 3, 4, 17, 18]. However, in the hybrid systems-based approach, a class of gene regulatory networks are limited to low-dimensional systems, because the computation time to solve the control problem is too long. In Boolean networks, dynamics such as interactions between genes are expressed by Boolean functions [15]. Although there is a criticism that a Boolean network is too simple as a model of gene regulatory networks, this model can be relatively applied to large-scale systems. Furthermore, since the behavior of gene regulatory networks is probabilistic by the effects of noise, a probabilistic Boolean network (PBN) has been proposed in [20].

Although a Boolean network is a class of discrete-time nonlinear systems and expresses the synchronous behavior, it is important to consider the asynchronous behavior. Asynchronous Boolean networks have been proposed in [12, 22]. In [12], we assume that the updating time of the concentration level of each gene

is given in advance. In [22], asynchronous Boolean networks are modeled by non-deterministic dynamical systems. Furthermore, the asynchronous behavior is expressed as the probabilistic behavior. However, in these two methods, the asynchronous behavior is not directly modeled.

On the other hand, a Petri net is well known as a model expressing the asynchronous behavior [25]. A Petri net is a class of directed bipartite graphs, in which the nodes represent transitions and places. The methods to express asynchronous Boolean networks as Petri nets have been proposed in [8, 21]. However, in these methods, the control input is not considered, and these methods cannot be directly applied to the control problem. The control input in biological networks has the following significance. For example, the value of the control input expresses whether a stimulus is given to a cell. Then the control input is designed to obtain the state trajectory that transits from the initial state to the desired one. So the control input can represent the current status of therapeutic interventions, which are realized by radiation, chemotherapy, and so on. In order to develop gene therapy technologies (see e.g., [16, 19]) in future, it is important to consider control methods of Boolean networks.

Thus in this paper, the optimal control problem of asynchronous Boolean networks modeled by Petri nets is discussed. First, based on the method proposed in [8] and the notation of external input places [13, 23], we propose a new method to transform asynchronous Boolean networks with control inputs into Petri nets with external input places. Furthermore, the obtained Petri net is transformed into a logical dynamical system, which is a class of linear systems with binary states and binary inputs. Next, the optimal control problem is formulated, and the biological significance is also discussed by using a simple example. Finally, the solution method of the optimal control problem is proposed. In the proposed solution method, the optimal control problem is reduced to an integer linear programming (ILP) problem. The proposed approach provides us a new control method of gene regulatory networks.

This paper is organized as follows. In Section 2, synchronous Boolean networks and asynchronous Boolean networks are introduced. In Section 3, Petri nets expressing asynchronous Boolean networks are derived. In Section 4, we explain the method to transform the obtained Petri net into a logical dynamical system. In Section 5, the optimal control problem is formulated, and in Section 6, a solution method is proposed. In Section 7, we conclude this paper.

Notation: Let $\mathcal{R}$ denote the set of real numbers. Let $\{0,1\}^{m\times n}$ denote the set of $m \times n$ matrices, which consists of elements 0 and 1. For the finite set M, let $|M|$ denote the number of elements. Let I_n and $0_{m\times n}$ denote the $n \times n$ identity matrix and the $m \times n$ zero matrix, respectively. For simplicity of notation, we sometimes use the symbol 0 instead of $0_{m\times n}$, and the symbol I instead of I_n. For a matrix M, let M^T denote the transpose of M.

2 Synchronous/Asynchronous Boolean Networks

First, we explain synchronous Boolean networks (SBNs).

Consider the following SBN:

$$x(k+1) = f_a(x(k)) \tag{1}$$

where $x \in \{0,1\}^n$ is the state (e.g., the concentration of genes), $k = 0, 1, 2, \ldots$ is the discrete time. $f_a : \{0,1\}^n \to \{0,1\}^n$ is a given Boolean function with logical operators such as AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). Since the SBN (1) is deterministic, $x(k+1)$ is uniquely determined for a given $x(k)$.

To consider the control problems, we add the control input to the SBN (1) as follows:

$$x(k+1) = f(x(k), u(k)) \tag{2}$$

where $u \in \{0,1\}^m$ is the control input, i.e., the value of u (e.g., the concentration of genes) can be arbitrarily given, and $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is a given Boolean function. The i-th element of the state x, the i-th element of the control input u and the i-th element of the Boolean function f are denoted by x_i, u_i and f_i, respectively. Also in the SBN (2), $x(k+1)$ is uniquely determined for given $x(k)$ and $u(k)$.

Next, we explain asynchronous Boolean networks (ABNs). Some methods for expressing ABNs have proposed so far [12, 22]. In this section, we explain a method in [22] for expressing ABNs as nondeterministic systems. A Petri net-based approach will be explained in Section 3. In the case that ABNs are expressed as nondeterministic systems, the behavior of ABNs is obtained by the union of the behaviors of the following n SBNs:

$$\Sigma_i : \begin{cases} x_i(k+1) = f_i(x(k), u(k)), \\ x_j(k+1) = x_j(k), \quad \forall j \in \{1, 2, \ldots, n\} \setminus \{i\}, \end{cases} \tag{3}$$

where $i = 1, 2, \ldots, n$.

We show an example of SBNs and ABNs.

Example 1. As a simple example, consider the following SBN of an apoptosis network [9]:

$$\begin{cases} x_1(k+1) = \neg x_2(k) \wedge u(k), \\ x_2(k+1) = \neg x_1(k) \wedge x_3(k), \\ x_3(k+1) = x_2(k) \vee u(k) \end{cases} \tag{4}$$

where the concentration level (high or low) of the inhibitor of apoptosis proteins (IAP) is denoted by x_1, the concentration level of the active caspase 3 (C3a) by x_2, and the concentration level of the active caspase 8 (C8a) by x_3. The concentration level of the tumor necrosis factor (TNF, a stimulus) is denoted by u, and is regarded as the control input.

In the case of synchronous Boolean dynamics, state transitions can be computed by directly using (4). For example, for $x(0) = [\ 1\ \ 1\ \ 1\ ]^T$ and $u(k) = 0$, we obtain $x(1) = [\ 0\ \ 0\ \ 1\ ]^T$. By computing the transition from each state, we obtain the state transition diagram in Fig. 1 (left). In Fig. 1 (left), the number assigned to each node denotes x_1, x_2, x_3 (elements of the state),

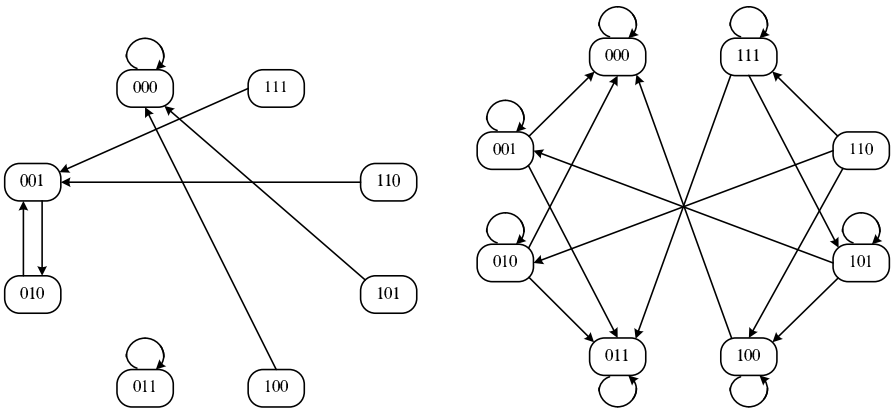


Fig. 1. (Left) State transition diagram of (4) and $u(k) = 0$, (Right) State transition diagram of (5), (6), (7) and $u(k) = 0$.

In the case of asynchronous Boolean dynamics, we consider the following three SBNs

$$\Sigma_1 : \begin{cases} x_1(k+1) = \neg x_2(k) \wedge u(k), \\ x_2(k+1) = x_2(k), \\ x_3(k+1) = x_3(k), \end{cases} \tag{5}$$

$$\Sigma_2 : \begin{cases} x_1(k+1) = x_1(k), \\ x_2(k+1) = \neg x_1(k) \wedge x_3(k), \\ x_3(k+1) = x_3(k) \end{cases} \tag{6}$$

$$\Sigma_3 : \begin{cases} x_1(k+1) = x_1(k), \\ x_2(k+1) = x_2(k), \\ x_3(k+1) = x_2(k) \vee u(k) \end{cases} \tag{7}$$

State transitions can be computed by using (5), (6), (7). For example, for $x(0) = [\ 1\ \ 1\ \ 1\ ]^T$ and $u(k) = 0$, we obtain $x(1) = \{[\ 0\ \ 1\ \ 1\ ]^T, [\ 1\ \ 0\ \ 1\ ]^T, [\ 1\ \ 1\ \ 1\ ]^T\}$. In a similar way, by computing the transition from each state, we obtain the state transition diagram in Fig. 1 (right).

Comparing the left figure with the right figure in Fig. 1, we see that a part of behaviors is clearly different. □

In this paper, ABNs are modeled by Petri nets, not multiple SBNs. By using Petri nets, we can consider several situations. For example, although each x_i is independently activated in (3), activation of combinations of x_i can be considered.

3 Transformation of Boolean Networks into Petri Nets

Using each Boolean function f_i in the SBN (2), consider to express an ABN as a Petri net. Based on a complementary-place transformation, a Petri net

expressing an ABN has been proposed in [8], but the control input has not been considered. In this paper, as an extension of the method in [8], we propose a modeling method of a Petri net expressing an ABN with the control input.

First, some notations are prepared. By $\mathcal{I}(j)$, $j = 1, 2, \ldots, n$, denote the state and the control input included in the Boolean function f_j. In the example of (4), we obtain $\mathcal{I}(1) = \{x_2, u\}$, $\mathcal{I}(2) = \{x_1, x_3\}$, and $\mathcal{I}(3) = \{x_2, u\}$. Next, we define a logical parameter $K_j(X) \in \{0, 1\}$, $X \subseteq \mathcal{I}(j)$, $j = 1, 2, \ldots, n$. If the value of each element included in X is '1', and the value of the other element is '0', then either $K_j(X) = 1$ or $K_j(X) = 0$ is determined. In the example of (4), we obtain

$$
\begin{aligned}
&K_1(\emptyset) = 0, \quad K_1(\{x_2\}) = 0, \quad K_1(\{u\}) = 1, \quad K_1(\{x_2, u\}) = 0,\\
&K_2(\emptyset) = 0, \quad K_2(\{x_1\}) = 0, \quad K_2(\{x_3\}) = 1, \quad K_2(\{x_1, x_3\}) = 0,\\
&K_3(\emptyset) = 0, \quad K_3(\{x_2\}) = 1, \quad K_3(\{u\}) = 1, \quad K_3(\{x_2, u\}) = 1.
\end{aligned}
$$

Next, consider to derive a Petri net expressing Boolean networks. In the derived Petri net, the number of places is given as $2(n + m)$, that is, for each x_i in (2), two places x_i and $\overline{x}_i$ are prepared. In a similar way, for each u_i in (2), two places u_i and $\overline{u}_i$ are prepared. $\overline{x}_i$ and $\overline{u}_i$ are called complementary places [8]. The number of transitions is given as $\sum_{i=1}^{n} 2^{|\mathcal{I}(i)|}$. In the example of (4), the number of transitions is given as $2^2 + 2^2 + 2^2 = 12$. From the property of Boolean networks, the following assumptions are made.

Assumption 1 *The maximum number of tokens in each place is equal to* 1.

Assumption 2 *A sum of the number of tokens in x_i (u_i) and that in $\overline{x}_i$ ($\overline{u}_i$) is equal to* 1.

In addition, suppose that u_i and $\overline{u}_i$ are given as an external input place [13, 23]. In u_i and $\overline{u}_i$, a token is arbitrary generated, but the above two assumptions must be satisfied.

Under the above preparations, we define a Petri net expressing an ABN. In [8], the Petri net expressing an ABN without the control input is defined. The following definition gives the Petri net expressing an ABN with the control input, and is an extension of the definition in [8].

Definition 1. *For a given SBN (2), the Petri net expressing an ABN is defined as follows:*

$$N_c = (P \cup P_c, T, Pre, Post) \tag{8}$$

where

- $P = \{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$ *is the set of places,*
- $P_c = \{u_1, \overline{u}_1, u_2, \overline{u}_2, \ldots, u_m, \overline{u}_m\}$ *is the set of external input places,*
- $T = \{t_{x_i, X}, i = 1, 2, \ldots, n, X \subseteq \mathcal{I}(i)\}$ *is the set of transitions,*
- $Pre : (P \cup P_c) \times T \to \{0, 1\}$ *is the mapping defining arcs between places and transitions,*
- $Post : T \times P \to \{0, 1\}$ *is the mapping defining arcs between transitions and transitions.*

The functions Pre and Post are defined as follows:
(i) Case of $x_i \notin \mathcal{I}(i)$ (x_i is not a self-regulator): For a given transition $t_{x_i,X}$, the following terms are defined (all the other terms are equal to zero):

$$\begin{aligned}
&Pre(x_i, t_{x_i,X}) = Post(t_{x_i,X}, \overline{x}_i) = 1 - K_i(X),\\
&Pre(\overline{x}_i, t_{x_i,X}) = Post(t_{x_i,X}, x_i) = 1 - K_i(X),\\
&Pre(x_j, t_{x_i,X}) = Post(t_{x_i,X}, x_j) = 1, \quad \forall x_j \in X,\\
&Pre(\overline{x}_j, t_{x_i,X}) = Post(t_{x_i,X}, \overline{x}_j) = 1, \quad \forall x_j \in \mathcal{I}(i) - X,\\
&Pre(u_j, t_{x_i,X}) = 1, \quad \forall u_j \in X,\\
&Pre(\overline{u}_j, t_{x_i,X}) = 1, \quad \forall u_j \in \mathcal{I}(i) - X.
\end{aligned}$$

(ii) Case of $x_i \in \mathcal{I}(i)$ (x_i is a self-regulator): Consider a given transition $t_{x_i,X}$. if $x_i \in X$, then only the case of $K_i(X) = 0$ is considered. Therefore, the following terms are defined:

$$\begin{aligned}
&Pre(x_i, t_{x_i,X}) = Post(t_{x_i,X}, \overline{x}_j) = 1,\\
&Pre(x_j, t_{x_i,X}) = Post(t_{x_i,X}, x_j) = 1, \quad \forall x_j \in X, \quad x_j \neq x_i,\\
&Pre(\overline{x}_j, t_{x_i,X}) = Post(t_{x_i,X}, \overline{x}_j) = 1, \quad \forall x_j \in \mathcal{I}(i) - X,\\
&Pre(u_j, t_{x_i,X}) = 1, \quad \forall u_j \in X,\\
&Pre(\overline{u}_j, t_{x_i,X}) = 1, \quad \forall u_j \in \mathcal{I}(i) - X.
\end{aligned}$$

if $x_i \notin X$, then only the case of $K_i(X) = 1$ is considered. Therefore, the following terms are defined:

$$\begin{aligned}
&Pre(\overline{x}_i, t_{x_i,X}) = Post(t_{x_i,X}, x_j) = 1,\\
&Pre(x_j, t_{x_i,X}) = Post(t_{x_i,X}, x_j) = 1, \quad \forall x_j \in X,\\
&Pre(\overline{x}_j, t_{x_i,X}) = Post(t_{x_i,X}, \overline{x}_j) = 1, \quad \forall x_j \in \mathcal{I}(i) - X, \quad x_j \neq x_i,\\
&Pre(u_j, t_{x_i,X}) = 1, \quad \forall u_j \in \mathcal{I}(i) - X,\\
&Pre(\overline{u}_j, t_{x_i,X}) = 1, \quad \forall u_j \in X.
\end{aligned}$$

In the above definition, a sum of the number of tokens in u_i and that in $\bar{u}_i$ becomes zero by firing some transition. In this case, to satisfy Assumption 1 and Assumption 2, a token is generated in either u_i or $\bar{u}_i$.

We show a simple example.

Example 2. Consider the following simple SBN:

$$\begin{cases} x_1(k+1) = x_2(k), \\ x_2(k+1) = u(k) \end{cases} \tag{9}$$

From $x_1(k+1) = x_2(k)$, we obtain $K_1(\emptyset) = 0$ and $K_1(\{x_2\}) = 1$. In a similar way, from $x_1(k+1) = u(k)$, we obtain $K_2(\emptyset) = 0$ and $K_1(\{u\}) = 1$. Then we consider four transitions $t_{x_1,\emptyset}$, $t_{x_1,\{x_2\}}$, $t_{x_2,\emptyset}$, and $t_{x_2,\{u\}}$. We denote these

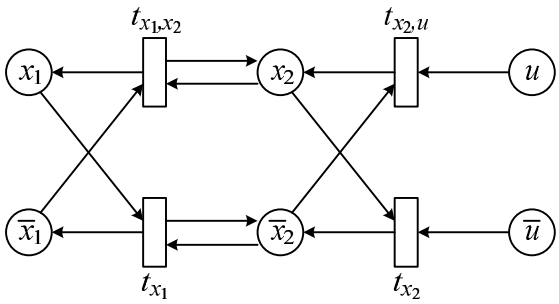


Fig. 2. Petri net expressing an ABN

transitions by t_{x_1}, t_{x_1,x_2}, t_{x_2}, and $t_{x_2,u}$, respectively. Then we obtain the Petri net in Fig. 2. In addition, *Pre* and *Post* in (8) are obtained as follows:

$$Pre = \begin{array}{c} \\ x_1 \\ \bar{x}_1 \\ x_2 \\ \bar{x}_2 \\ u \\ \bar{u} \end{array} \begin{array}{c} \begin{array}{cccc} t_{x_1} & t_{x_1,x_2} & t_{x_2} & t_{x_2,u} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}, \quad Post = \begin{array}{c} \\ t_{x_1} \\ t_{x_1,x_2} \\ t_{x_2} \\ t_{x_2,u} \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & \bar{x}_1 & x_2 & \bar{x}_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}.$$

Suppose that one token is included in place $\bar{x}_1$, $\bar{x}_2$, and u. Then the transition $t_{x_2,u}$ may fire. If the transition $t_{x_2,u}$ fire, then one token is moved from $\bar{x}_2$ and u to x_2. A pair of x_2 and $\bar{x}_2$ satisfies Assumption 1 and Assumption 2, but a pair of u and $\bar{u}$ does not satisfy Assumption 2. So one token must be added in either u and $\bar{u}$ with fire. □

4 Transformation of Petri Nets into Logical Dynamical Systems

To consider the optimal control problem, it is desirable to transform a Petri net (8) into some linear form. In this section, based on a framework on modeling of hybrid dynamical systems [5], logical dynamical systems expressing Petri nets are derived.

Logical dynamical systems are given as a pair of linear state equations and linear inequality constraints with binary state variables and binary control input variables. A general form of logical dynamical systems is defined as follows.

Definition 2. *A logical dynamical system is given as*

$$x(k+1) = Ax(k) + Bv(k), \tag{10}$$

$$Cx(k) + Dv(k) \leq E \tag{11}$$

where $x(k) \in \{0,1\}^{n_d}$ *is the state variable, and* $v(k) \in \{0,1\}^{m_d}$ *is the input variable including auxiliary variables,*

In modeling of hybrid dynamical systems, a mixed logical dynamical (MLD) system [5] is well known. The MLD system can be derived by replacing $x(k) \in \{0,1\}^{n_d}$ and $v(k) \in \{0,1\}^{m_d}$ in (10), (11) with $x(k) \in \mathcal{R}^{n_c} \times \{0,1\}^{n_d}$ and $v(k) \in \mathcal{R}^{m_c} \times \{0,1\}^{m_d}$. Since a Petri net is a class of discrete event systems, the MLD system is not used, and the logical dynamical system (10), (11) is used.

Let us consider to transform the Petri net (8) into the logical dynamical system (10), (11). Some notations are prepared. By $x_i(k), \bar{x}_i(k), u_i(k), \bar{u_i}(k) \in \{0,1\}$, denote existence or non-existence of a token in place $x_i, \bar{x}_i, u_i, \bar{u}_i \in \{0,1\}$ at time k. k may be regarded as the k-th firing in a firing sequence. Since $x_i(k), \bar{x}_i(k), u_i(k), \bar{u}_i(k)$ are binary variables, Assumption 1 satisfies. To satisfy Assumption 2, $x_i(k) + \bar{x}_i(k) = 1$ and $u_i(k) + \bar{u}_i(k) = 1$ are imposed. Next, by $t_{x_i,X_j}(k) \in \{0,1\}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, 2^{|\mathcal{I}(i)|}$, denote fire in the transition t_{x_i,X_j}. If $t_{x_i,X_j}(k) = 1$, then the transition t_{x_i,X_j} fires at time k. Otherwise, t_{x_i,X_j} does not fire. By using these notations, Petri nets (8) is transformed into logical dynamical systems.

First, a simple example is shown.

Example 3. Consider the Petri net in Fig. 2. From the property of fire, we obtain the following system expressing the Petri net in Fig. 2:

$$\begin{aligned}
x_1(k+1) &= t_{x_1,x_2}(k)\bar{x}_1(k)x_2(k) + (1 - t_{x_1,x_2}(k))x_1(k) \\
&\quad -t_{x_1}(k)x_1(k)\bar{x}_2(k), && (12)\\
\bar{x}_1(k+1) &= t_{x_1}(k)x_1(k)\bar{x}_2(k) + (1 - t_{x_1}(k))\bar{x}_1(k) \\
&\quad -t_{x_1,x_2}(k)\bar{x}_1(k)x_2(k), && (13)\\
x_2(k+1) &= t_{x_1,x_2}(k)\bar{x}_1(k)x_2(k) + t_{x_2,u}(k)\bar{x}_2(k)u(k) \\
&\quad +(1 - t_{x_1,x_2}(k) - t_{x_2,u}(k))x_2(k) - t_{x_2}(k)x_2(k)\bar{u}(k), && (14)\\
\bar{x}_2(k+1) &= t_{x_1}(k)x_1(k)\bar{x}_2(k) + t_{x_2}(k)x_2(k)\bar{u}(k) \\
&\quad +(1 - t_{x_1}(k) - t_{x_2}(k))\bar{x}_2(k) \\
&\quad -t_{x_1}(k)x_1(k)\bar{x}_2(k) - t_{x_2,u}(k)\bar{x}_2(k)u(k). && (15)
\end{aligned}$$

If the number of firing transitions at each time is limited to 1, then the inequality condition

$$t_{x_1}(k) + t_{x_1,x_2}(k) + t_{x_2}(k) + t_{x_2,u}(k) \leq 1 \qquad (16)$$

is imposed. Then $x_1(k), \bar{x}_1(k), x_2(k), \bar{x}_2(k) \in \{0,1\}$, $x_1(k) + \bar{x}_1(k) = 1$, and $x_2(k) + \bar{x}_2(k) = 1$ hold thanks to the condition (16), $u(k), \bar{u}(k) \in \{0,1\}$, $u(k) + \bar{u}(k) = 1$, and the initial condition $x_1(0), \bar{x}_1(0), x_2(0), \bar{x}_2(0) \in \{0,1\}$, $x_1(0) + \bar{x}_1(0) = 1$, $x_2(0) + \bar{x}_2(0) = 1$. The system (12)–(15) is a nonlinear system, and can be linearized by using Lemma 1 in Appendix A. For example, $z_1(k) = t_{x_1,x_2}(k)\bar{x}_1(k)x_2(k)$ is equivalent to

$$\begin{aligned}
&t_{x_1,x_2}(k) + \bar{x}_1(k) + x_2(k) - z_1(k) \leq 2, \\
&-t_{x_1,x_2}(k) - \bar{x}_1(k) - x_2(k) + 3z_1(k) \leq 0.
\end{aligned}$$

By applying Lemma 1 to the other terms, we can obtain the logical dynamical system expressing the Petri net in Fig. 2. □

From this example, we see that the Petri net (8) can be expressed by the logical dynamical system (10), (11). Furthermore, when the Petri net (8) is expressed by the logical dynamical system (10), (11), variables x, v are given as

$$
\begin{aligned}
x(k) &= [\ x_1(k)\ \ \bar{x}_1(k)\ \ x_2(k)\ \ \bar{x}_2(k)\ \ \cdots\ \ x_n(k)\ \ \bar{x}_n(k)\ ]^T, \\
v(k) &= [\ U(k)\ \ T(k)\ \ Z(k)\ ]^T, \\
U(k) &= [\ u_1(k)\ \ \bar{u}_1(k)\ \ u_2(k)\ \ \bar{u}_2(k)\ \ \cdots\ \ u_m(k)\ \ \bar{u}_m(k)\ ]^T, \\
T(k) &= \left[\ t_{x_1,X_1}(k)\ \ \cdots\ \ t_{x_1,X_{2^{|\mathcal{I}(1)|}}}(k)\ \ \cdots\ \ t_{x_n,X_1}(k)\ \ \cdots\ \ t_{x_n,X_{2^{|\mathcal{I}(n)|}}}(k)\ \right]^T
\end{aligned}
$$

where $Z(k)$ is a auxiliary binary variable obtained by applying Lemma 1. Matrices/vectors A, B, C, D, E in (10), (11) can be derived from $Pre, Post$ in the Petri net (8) and Lemma 1.

Remark 1. One of the simple methods for modeling of ABNs is to express ABNs as switched systems with 2^n subsystems. 2^n subsystems are derived by all combinations of Boolean functions $f_1, f_2, \ldots, f_n$ in (2). In the case of (9) in Example 2, the following $2^n = 4$ subsystems:

$$
\begin{aligned}
&\Sigma_1 : x_1(k+1) = x_1(k), \quad x_2(k+1) = x_2(k), \\
&\Sigma_2 : x_1(k+1) = x_2(k), \quad x_2(k+1) = x_2(k), \\
&\Sigma_3 : x_1(k+1) = x_1(k), \quad x_2(k+1) = u(k), \\
&\Sigma_4 : x_1(k+1) = x_2(k), \quad x_2(k+1) = u(k)
\end{aligned}
$$

are obtained. From these subsystems, we can obtain the following system expressing an ABN:

$$x_1(k+1) = (\delta_1(k) + \delta_3(k))x_1(k) + (1 - \delta_1(k) - \delta_3(k))x_2(k), \tag{17}$$

$$x_2(k+1) = (\delta_1(k) + \delta_2(k))x_2(k) + (1 - \delta_1(k) - \delta_2(k))u(k) \tag{18}$$

where $\delta_1(k), \delta_2(k), \delta_3(k)$ are binary variables satisfying $\delta_1(k)+\delta_2(k)+\delta_3(k) \leq 1$, and correspond to $\Sigma_1, \Sigma_2, \Sigma_3$, respectively. Furthermore, $z_1 := \delta_1 x_1$, $z_2 := \delta_3 x_1$, $z_3 := \delta_1 x_2$, $z_4 := \delta_3 x_2$, $z_5 := \delta_2 x_2$, $z_6 := \delta_1 u$, and $z_7 := \delta_2 u$ are defined, and Lemma 1 is applied to $z_1, z_2, \ldots, z_7$. Thus we can obtain the logical dynamical system (10), (11). This method is called here a direct approach.

Comparing (17), (18) with (12)–(15), we see that the system (17), (18) is simpler than the system (12)–(15). However, for general cases, this fact does not hold. Here, we focus on the dimension of binary variables to switch Boolean functions. In (12)–(15), this dimension corresponds to the dimension of binary variables assigned to transitions, and is given as 4. In (17), (18), this dimension is given as 3. In general, in the direct approach, the dimension of binary variables to switch Boolean functions is given as $2^n - 1$. In the proposed Petri net-based approach, this dimension, i.e., $|T(k)|$ is given as $\sum_{i=1}^{n} 2^{|\mathcal{I}(i)|}$. In real gene regulatory networks, it is well known that $|\mathcal{I}(i)|$ is relatively smaller than n (see e.g., [2]). For example, in the case of $n = 10$ and $|\mathcal{I}(i)| = 3$, $2^n - 1 = 1023$ and $\sum_{i=1}^{n} 2^{|\mathcal{I}(i)|} = 80$ are obtained. Thus, in modeling of real gene regulatory networks, it is not appropriate to use the direct approach, and the proposed Petri net-based method provides us a simpler modeling method of ABNs. □

5 Optimal Control Problem

For the logical dynamical system (10), (11) expressing the Petri net (8), consider the following optimal control problem.

Problem 1. For the logical dynamical system (10), (11) expressing the Petri net (8), suppose that the initial state $x(0) = x_0$ satisfying Assumption 1 is given. Then find an input sequence $v(0), v(1), \ldots, v(N-1)$ minimizing the linear cost function

$$J = \sum_{k=0}^{N-1} \{Qx(k) + Rv(k)\} + Q_f x(N) \tag{19}$$

where $Q, Q_f \in \mathcal{R}^{1 \times n_d}$, $R \in \mathcal{R}^{1 \times m_d}$ are weighting vectors whose element is a non-negative real number.

For simplicity of discussion, a linear function with respect to x and u is considered as a cost function, but a quadratic cost function may be used. In addition, using the offset vector $x_d \in \{0,1\}^{n_d}$ and $v_d \in \{0,1\}^{m_d}$, $x(k)$ and $v(k)$ may be replaced to $\hat{x}(k) := x(k) - x_d$ and $\hat{v}(k) := v(k) - v_d$. Then it is necessary that the cost function (19) is also replaced to $J = \sum_{i=0}^{N-1} \{Q|\hat{x}(i)| + R|\hat{v}(i)|\} + Q_f|\hat{x}(N)|$. In addition, N must be determined according to a given biological network. Although a longer N is desirable, the computation time to solve Problem 1 must be also considered. For a small N, Problem 1 may be repeatedly solved at each time. This policy is well known as model predictive control [6].

In Problem 1, we assume that an input sequence $v(0), v(1), \ldots, v(N-1)$ is arbitrarily determined. However, there is a possibility that a given biological system does not satisfy this assumption. Then suppose that some candidates of input sequences are given. In Problem 1, the optimal input sequence minimizing the cost function (19) is selected among the set of the candidates $\mathcal{B} \subseteq \{0,1\}^{m_d N}$. This extension is easy. In this sense, Problem 1 can be applied to optimal control of asynchronous Boolean networks such that the updating time of each state is given in advance [12]. Of course, this problem can also be applied to optimal control of SBNs. Thus Problem 1 includes several situations.

Next, we show an example for setting weighting vectors from the biological viewpoint.

Example 4. Consider the Boolean network expressing an apoptosis network in Example 1 again. From (4), we obtain the Petri net (8) with 6 places, 2 external input places, and 12 transitions. In addition, from the obtained Petri net, we obtain the logical dynamical system (10), (11). For the obtained logical dynamical system, we consider to find a control strategy such that a stimulus is not applied as much as possible, and cell survival is achieved. $u(k) = 0$ implies that a stimulus is not applied to the system, and $x_1(k) = 1$, $x_2(k) = 0$ express cell survival [9]. Then as one of appropriate cost functions, we can consider the following cost function

$$J = \sum_{k=0}^{N-1} \{10|x_1(k) - 1| + 10|x_2(k) - 0| + u(k)\}$$

$$+100|x_1(N) - 1| + 100|x_2(N) - 0|.$$

By the appropriate coordinate transformation, this cost function can be rewritten as the form of (19). See also [10–12] for biological examples on the optimal control problems. □

6 Reduction to an Integer Linear Programming Problem

Finally, let us consider to reduce Problem 1 to an integer linear programming (ILP) problem.

Problem 1 can be rewritten by using (10), (11). First, by using

$$x(k) = A^k x_0 + \sum_{i=1}^{k} A^{i-1} B v(k-i)$$

obtained from the state equation (10), we obtain

$$\bar{x} = \bar{A} x_0 + \bar{B} \bar{v} \tag{20}$$

where $\bar{x} := [\ x^T(0)\ \ x^T(1)\ \ \cdots\ \ x^T(N)\ ]^T$, $\bar{v} := [\ v^T(0)\ \ v^T(1)\ \ \cdots\ \ v^T(N-1)\ ]^T$ and

$$\bar{A} := \begin{bmatrix} I_{n_d} \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \bar{B} := \begin{bmatrix} 0_{n_d \times m_d} & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix}.$$

Furthermore, from the linear inequality (11), we obtain

$$\bar{C} \bar{x} + \bar{D} \bar{v} \leq \bar{E} \tag{21}$$

where

$$\begin{aligned} \bar{C} &:= [\ \text{block-diag}(C, C, \ldots, C)\ \ 0\ ], \\ \bar{D} &:= \text{block-diag}(D, D, \ldots, D), \\ \bar{E} &:= \left[\ E^T\ \ E^T\ \ \cdots\ \ E^T\ \right]^T. \end{aligned}$$

Next, the cost function (19) can also be rewritten as

$$J = \bar{Q} \bar{x} + \bar{R} \bar{v} \tag{22}$$

where $\bar{Q} := [\ Q\ \ \cdots\ \ Q\ \ Q_f\ ]$ and $\bar{R} := [\ R\ \ \cdots\ \ R\ ]$. By substituting (20) into (21) and (22), we obtain the following theorem.

Theorem 1. *Problem 1 is equivalent to the following ILP problem.*

Problem A:

$$\begin{aligned} &\text{find} && \bar{v} \in \{0,1\}^{m_d N}, \\ &\min && (\bar{R} + \bar{Q}\bar{B})\bar{v} + \bar{Q}\bar{A}x_0, \\ &\text{subject to} && \begin{bmatrix} \bar{C}\bar{B} + \bar{C} \\ -\bar{L}\bar{W} \end{bmatrix} \bar{v} \leq \begin{bmatrix} \bar{E} - \bar{C}\bar{A}x_0 \\ -\ln \rho \end{bmatrix}. \end{aligned}$$

Problem A can be solved by using a suitable ILP solver such as IBM ILOG CPLEX Optimizer [24].

7 Conclusion

In this paper, we have discussed optimal control of asynchronous Boolean networks with control inputs. First, we have proposed a method to transform Boolean networks with control inputs into Petri net with external input places. Next, after the obtained Petri net is transformed into a logical dynamical system, the optimal control problem has been formulated. This problem is a general formulation including several biological situations. Finally, the optimal control problem has been reduced to an integer linear programming problem. The proposed approach will be effective for control of several biological systems modeled by Boolean networks.

One of the future works is to apply the proposed approach to biological Boolean networks. From the practical viewpoint, an extension to probabilistic Boolean networks is also important. In addition, for large-scale Boolean networks, the computation time to solve the problem will be long. So it is significant to consider to reduce the computation time to solve the problem. Then one of methods to overcome this difficulty is to use a SAT (satisfiability problem) solver such as zChaff [26]. It is also important to consider approximate solution methods.

This work was supported by Grant-in-Aid for Young Scientists (B) 23760387 and Scientific Research (C) 21500009.

A Linearization of The Product of Binary Variables

The product of binary variables can be linearized by using the following lemma [7].

Lemma 1. *Suppose that binary variables $\delta_j \in \{0,1\}$, $j \in \mathcal{J}$ are given, where $\mathcal{J}$ is some index set. Then $z = \prod_{j\in\mathcal{J}} \delta_j$ is equivalent to the following linear inequalities*

$$\sum_{j\in\mathcal{J}} \delta_j - z \leq |\mathcal{J}| - 1, \quad -\sum_{j\in\mathcal{J}} \delta_j + |\mathcal{J}|z \leq 0$$

where $|\mathcal{J}|$ is the cardinality of $\mathcal{J}$.

References

1. T. Akutsu, M. Hayashida, W.-K. Ching, and M. K. Ng, Control of Boolean networks: Hardness results and algorithms for tree structured networks, *Journal of Theoretical Biology*, vol. 244, pp. 670–679, 2007.
2. M. Andrecut, S. A. Kauffman, and A. M. Madni, Evidence of Scale-Free Topology in Gene Regulatory Network of Human Tissues, *Int'l Journal of Modern Physics C*, vol. 19, no. 2, pp. 283–290, 2008.
3. S. Azuma, E. Yanagisawa, and J. Imura, Controllability Analysis of Biosystems Based on Piecewise Affine Systems Approach, *IEEE Trans. on Automatic Control*, vol. 53, no. 1, pp. 139–152, 2008.
4. C. Belta, J. Schug, T. Dang, V. Kumar, G. J. Pappas, H. Rubin, and P. Dunlap, Stability and reachability analysis of a hybrid model of luminescence in the marine bacterium *Vibrio fischeri*, *Proc. 40th IEEE Conf. on Decision and Control*, pp. 869–874, 2001.
5. A. Bemporad and M. Morari, Control of systems integrating logic, dynamics, and constraints. *Automatica*, Vol. 35, pp. 407–427, 1999.
6. E. F. Camacho, C. Bordons, *Model Predictive Control*, 2nd Edition, Springer, 2004.
7. T. M. Cavalier, P. M. Pardalos, and A. L. Soyster, Modeling and integer programming techniques applied to propositional calculus, *Computer & Operations Research*, vol. 17, no. 6, pp. 561–570, 1990.
8. C. Chaouiya, E. Remy, P. Ruet, and D. Thieffry, Qualitative Modelling of Genetic Networks: from Logical Regulatory Graphs to Standard Petri Nets, *25th Int'l Conf. on Application and Theory of Petri Nets*, *LNCS 3099*, pp. 137–156, 2004.
9. M. Chaves, Methods for Qualitative Analysis of Genetic Networks, *Proc. European Control Conference 2009*, pp. 671–676, 2009.
10. A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, External Control in Markovian Genetic Regulatory Networks, *Machine Learning*, vol. 52, pp. 169–191, 2003.
11. A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, External control in Markovian genetic regulatory networks: the imperfect information case, *Bioinfomatics*, vol. 20, pp. 924–930, 2004.
12. B. Faryabi, J.-F. Chamberland, G. Vahedi, A. Datta, and E. R. Dougherty, Optimal Intervention in Asynchronous Genetic Regulatory Networks, *IEEE Journal of Selected Topics in Signal Processing*, Vol. 2, No. 3, pp. 412–423, 2008.
13. A. Ichikawa and K. Hiraishi, Analysis and Control of Discrete Event Systems Represented by Petri Nets, *Discrete Event Systems: Models and Applications*, *LNCIS 103*, pp. 115–134, 1988.
14. H. D. Jong, Modeling and Simulation of Genetic Regulatory Systems: A Literature Review, *Journal of Computational Biology*, vol. 9, pp. 67–103, 2002.
15. S. A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.
16. M.-S. Kim *et al.*, DNA demethylation in hormone-induced transcriptional derepression, *Nature*, vol. 461, pp. 1007–1012, 2009.
17. K. Kobayashi, J. Imura, and K. Hiraishi, Polynomial-Time Algorithm for Controllability Test of a Class of Boolean Biological Networks, *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2010, Article ID 210685, 12 pages, 2010.
18. C. J. Langmead and S. K. Jha, Symbolic Approaches to Finding Control Strategies in Boolean Networks, *Journal of Bioinformatics and Computational Biology*, vol. 7, no. 2, pp. 323–338, 2009.

19. H. Santos-Rosa and C. Caldas, Chromatin modifier enzymes, the histone code and cancer, *European Journal of Cancer*, vol. 41, pp. 2381–2402, 2005.
20. I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks, *Bioinformatics*, vol. 18, pp. 261–274, 2002.
21. L.J. Steggles, R. Banks, and A. Wipat, Modelling and Analysing Genetic Networks: From Boolean Networks to Petri Nets, *Int'l Conf. on Computational Methods in Systems Biology*, *LNBI 4210*, pp. 127–141, 2006.
22. L. Tournier and M.Chaves, Uncovering operational interactions in genetic networks using asynchronous Boolean dynamics, *Journal of Theoretical Biology*, vol. 260, pp. 196–209, 2009.
23. T. Ushio, Maximally Permissive Feedback and Modular Control Synthesis in Petri Nets with External Input Places, *IEEE Trans on Automatic Control*, Vol. 35, No. 7, pp. 844–848, 1990.
24. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
25. http://www.informatik.uni-hamburg.de/TGI/PetriNets/
26. http://www.princeton.edu/~chaff/zchaff.html

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.21-35

GReg : a domain specific language for the modeling of genetic regulatory mechanisms

Nicolas Sedlmajer, Didier Buchs, Steve Hostettler,
Alban Linard, Edmundo Lopez, and Alexis Marechal

Université de Genève, 7 route de Drize, 1227 Carouge, Switzerland

Abstract. Chemical and biological systems have similarities with *IT-systems* as they can be observed as sequences of events. Most available tools propose simulation frameworks to explore biological pathways (*i.e.,* sequences of events). Simulation only explores a few of the most probable pathways in the system. On the contrary, techniques such as model checking, coming from IT-systems analysis, explore all the possible behaviors of the modeled systems, thus helping to identify interesting pathways. A main drawback from most model checking tools in the life sciences domain is that they take as input a language designed for computer scientists, that is not easily understood by non-expert users. We propose in this article an approach based on Domain Specific Languages. It provides a comprehensible language to describe the system while allowing the use of complex and powerful underlying model checking techniques.

1 Introduction

Because of their stochastic and combinatorial nature, many biological systems such as cellular and supra-cellular interactions are very hard to investigate. Current practice is mainly limited to the use of *in vivo* and *in vitro* experiments. Investigation through formal models of biological systems is currently a rather restricted research field, unlike what has been done in other natural sciences such as chemistry and physics. There is clearly an emerging field of research where future experiments can be partially performed *in silico*, *i.e.,* by means of techniques from computer science. One of the main approaches of biological modeling is the so-called regulatory networks [17,5]. The main idea of biological modeling according to the regulatory network approach is to model interbiological reactions through a set of interdependent biological rules. This can be seen as a set of discrete modules having strong interconnections. The occurrence of interesting events in the biological system can be represented as logical properties expressed on the state of these modules. This is very similar to the kind of properties computer scientists validate on hardware and software systems (deadlocks, error states,...).

Among the tools available in this domain, the main analysis approach for regulatory networks is *simulation*. Simulation is generating and analyzing a limited sample of possible system behaviors. This technique is not convenient when the

main purpose of the research is to look for rare or abnormal behaviors (*e.g.,* cancer). The main approach in this case is to use *model checking* instead of simulation. Model checking consists in generating and analyzing the complete set of possible states of the system. Naturally, this technique suffers from the drawback of the enormous number of possible states of biological systems.

It is interesting to note that this problem is well-known to the model checking community in computer science, where it is called the *state space explosion* [18]. There is a parallel between cellular interactions and software systems in that the state space explosion is mainly due to their concurrent nature. Therefore, we can apply techniques that have been developed for the model checking of hardware and software systems to biological interactions. Approaches based on a symbolic encoding of the state space are particularly well-suited for this [4,9].

In this paper we show a work in progress in our group. We present Gene Regulation Language (GReg), our first attempt to build a framework for modeling and analyzing biological systems based on formal modeling and reasoning. Advanced techniques for defining Domain Specific Languages, giving their semantics and analyzing them using symbolic model checking are presented. First, Section 2 describes precisely the biological domain considered by GReg, then Section 3 outlines the state of the current research in the field and Section 4 describes in detail the creation and usage of Domain Specific Languages. The following two chapters describe GReg itself. Section 5 describes the language designed for expressing biological mechanisms and the corresponding queries, and Section 6 provides a simple example taken from the literature. Finally, Section 7 concludes the article and discusses the future research perspectives in this area.

2 Chemical and biological models covered by GReg

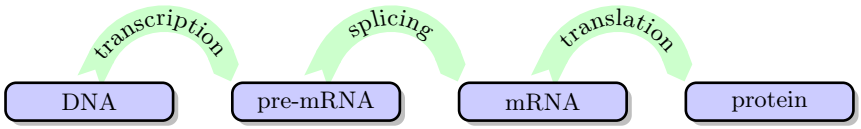


Fig. 1: DNA to protein process

The purpose of GReg is to describe genetic regulatory mechanisms controlling the DNA to protein process (Figure 1). This process comprises three steps: transcription, splicing and translation. The regulation of each step can be modeled using standard chemical reactions, presented in Section 2.1. For the transcription initiation we use the genetic regulatory mechanism model, presented in Section 2.2. Finally we define a cell network in Section 2.3. We separate these three domain models to clearly distinguish chemical from biological concepts.

We use the same definition of molecule level as presented in [17], if a molecule has n distinct actions, we define $(n + 1)$ levels. This allows us to use a discrete formalism to efficiently model the gene expression [17], which is in fact the

concentration of the gene products. The lowest level is the lowest transcription rate of a gene.

2.1 Chemical compartment model

A chemical compartment $\kappa = \langle M_\kappa, R \rangle \in K$, is composed of a non-empty set of molecules ($\varnothing \neq M_\kappa \subseteq Molecules$) and a set of reactions ($R \subseteq Reactions$). Two compartments may be separated by a membrane (*i.e.,* selective barrier), thus allowing molecule transfer between them.

A chemical reaction $\rho = \langle Re, Pr, Ca, k, type_\rho \rangle \in R$, where $Re, Pr, Ca \subseteq \mathcal{M}(M_\kappa) \times K$, the reactants ($Re$), products ($Pr$) and catalysts ($Ca$) are defined as the association of a multiset of one molecule ($\mathcal{M}(M_\kappa)$) with a given compartment ($\kappa \in K$).

Catalysts (Ca) have the particularity to appear as reactants and products in a chemical reaction. We have chosen to define the catalysts in a distinct set, therefore no catalyst can appear as reactant or product:
$\forall m \in M_\kappa, m \in Ca \implies m \notin Re \wedge m \notin Pr$.

A reaction may be either irreversible or reversible, $type_\rho \in \{irr, rev\}$. In reversible reactions an equilibrium constant (k) is defined with the ratio of both direction rates.

2.2 Genetic regulatory mechanism model

A genetic regulatory mechanism $\mu = \langle \Gamma_\mu, C \rangle$, is composed of a non-empty set of genes (Γ_μ). Genes are organized into one or more chromosomes (C). A genetic regulatory mechanism is contained in a chemical compartment, this specification will be presented in Section 2.3.

A chromosome $c = \lambda_1, \ldots \lambda_n \in C$ is a sequence of loci. A gene may have different version (*i.e.,* alleles) at a given chromosome location (*i.e.,* locus). And a locus $\lambda = \langle \Gamma_\lambda \rangle$ defines a non-empty set of genes ($\varnothing \neq \Gamma_\lambda \subseteq \Gamma_\mu$) that are located at a given locus. Then the set of all possible chromosomes in a mechanism is the Cartesian product of the set of genes at each locus:
$Chromosomes = \Gamma_{\lambda_1} \times \cdots \times \Gamma_{\lambda_n}$.

A gene $\gamma = \langle M_\gamma, \Sigma \rangle \in \Gamma_\mu$ is a portion of DNA that codes for at least one molecule ($\varnothing \neq M_\gamma \subseteq M_\kappa$), and may contain some regulation sites ($\Sigma \subseteq Sites$).

A regulation site $\sigma = \langle M_\sigma, type_\sigma \rangle \in \Sigma$ defines the non-empty set of regulatory molecules ($\varnothing \neq M_\sigma \subseteq M_\kappa$) associated to the regulation site σ of a given gene. Note that when using anti-termination sites, genes order matters, therefore chromosomes must be defined.

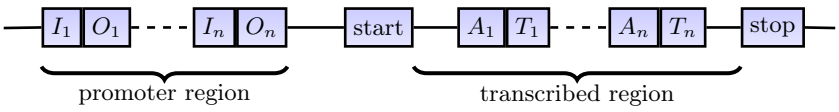


Fig. 2: Idealized gene structure

Our idealized gene structure (Figure 2) is composed of two regions: promoter and transcribed. Note that the exact position in the gene of each regulatory site is not specified, *i.e.,* we are mainly interested in its regulatory role. The type of a regulation site is $type_\sigma \in \{\mathbf{I},\mathbf{O},\mathbf{A},\mathbf{T}\}$.

Initiation (I) the portion of DNA where bound activators increase the rate of the transcription process. It is located in the promoter region;

Operator (O) the portion of DNA where bound repressors block the transcription process. It is located in the promoter region;

Anti-termination (A) the portion of DNA where activators continue the transcription process. It is located in the transcribed region. These sites may also allow the transcription of the next gene;

Termination (T) (also called attenuator) the portion of DNA where repressors stop the transcription process. It is located in the transcribed region; thus, it produces a reduced RNA.

2.3 Cell network

The cell network model is designed to model the interactions of different cells with their environment and also with their inner components (*i.e.,* organelles). This model authorizes the construction of currently not observed cells, *e.g.,* prokaryote with nucleus, eukaryote with multiple nuclei, *etc.* The validity of the specification is delegated to the user (*i.e.,* domain expert).

This model defines three chemical compartments, therefore they inherit both sets M_κ and R.

Organelle, $\omega = \langle M_\kappa, R, \mu \rangle$ is the lowest compartment in the compartment hierarchy. An organelle may contain a mechanism (μ), *e.g.,* nucleus, mitochondrial DNA, *etc.*

Cell, $\phi = \langle M_\kappa, R, \mu, \Omega \rangle$ contains a possibly empty set of organelles (Ω). The model of a prokaryote cell would define a mechanism (μ), by cons an eukaryote cell would define instead an organelle with a mechanism (*i.e.,* nucleus).

Network, $\nu = \langle M_\kappa, R, \Phi \rangle$ represents the environment and contains one or more cells (Φ).

3 Related work

In this section we compare three well-known tools with our approach. We first define a few criteria such as the kind of analysis, the supported formalism and the supported exchange format. Table 1 presents a summary of the resulting comparison.

Domain language To be productive, the syntax of the input language should be as close as possible to the actual domain of the user. This input language can be textual (like in tools that use Systems Biology Markup Language (SBML) [8]) or graphical (like Systems Biology Graphical Notation (SBGN) [11]).

Simulation & Model Checking Although there are many tools adapted to biological process design and simulation, only a few of them allow exhaustive exploration of the state space. While simulation is very useful during model elaboration, an exhaustive search may help to discover pathological cases that would have never been explored by simulation.

Discrete & continuous Continuous models are closer to the real biological systems than discrete models, but unlike the latter they are not adapted for model checking techniques. Discrete formalisms allow a complete exploration of the state space while preserving the qualitative properties of the system, as mentioned in [17].

Exchange format The supported interchange format is an important feature as it allows us to bridge the gap between different tools and therefore enables the user to use the most adapted tool to hand. SBML is a common interchange format based on XML. It is used to describe biochemical reactions, gene regulation and many other topics.

Cell Illustrator [16] is an example of a commercial simulation tool for continuous and discrete domains. The graphical formalism is based on PN, called Hybrid Functional Petri Nets with extensions (HFPNe), which adds the notions of continuous and generic processes and quantities [12]. The XML-based exchange file format used in Cell Illustrator is called CSML.

Gene Interaction Network simulation a.k.a. GinSim [14] is a tool for the modeling and simulation of genetic regulatory networks. It models genetic regulatory networks based on a discrete formalism [6,13]. These models are stored using the XML-based format *ginml*. The simulation computes a state transition graph representing the dynamical behavior network. GINsim uses a graphical Domain Specific Language (DSL) called Logical Regulatory Graph (LRG) [5]. Models in LRG are graphs, where nodes are regulatory components (*i.e.,* molecules and genes) and arcs are regulatory interactions (*i.e.,* activation and repression) between the nodes.

Cytoscape [7,15] is an open source software platform for visualizing complex networks and integrating these with any type of attribute data. Cytoscape supports many file formats including PSI-MI and SBML. It has the advantage of adding features through a plug-in system. Many plug-ins are available for various domains such as biology, bioinformatics, social network analysis and the semantic web. Over 100 plug-ins are listed on the official website.

4 DSL approach

Model checking involves verifying whether a property holds on the whole set of possible states of a given model. To generate this complete state space, the model must be expressed in a formal language intended for this operation, like Petri Nets (PNs). This makes the model checking approach impractical for people who do not master these formal languages. We propose using DSLs for this purpose.

Tool	Cell Illustrator	GINsim	Cytoscape	GReg
Domain language	✓	✓	✓	✓
Simulation	✓	×	✓	×
State space	×	✓	×	✓
Model checking	×	×	×	✓
Discrete	✓	✓	✓	✓
Continuous	✓	×	✓	×
Exchange format	CSML	GINML	SBML,...	GReg

Table 1: Tool comparison table.

A DSL is a programming or specification language tailored for a given domain; it presents a reduced set of instructions closely related to this domain. Using a DSL has two main objectives. First, learning the language should be easy for someone with enough knowledge about the domain, even if this person does not have previous knowledge of other languages. Second, the number of errors made by a novice user should be drastically reduced as the expressivity of the language is reduced to the minimum.

The DSL semantics are defined by transformation into a target language, which is a formal language where complex operations (like model checking) can be performed. Usually, the scope of the target language is broader than the scope of the DSL. This allows using the same target language and its associated tools for different DSLs. Moreover, while creating a new DSL, it is often possible to use an already-existing language as a target, thus facilitating the language creation process. The results obtained in the target language are translated again into the DSL and returned to the user. This process is described in Figure 3. After the creation of the initial model, all the following steps must be fully automatic, to hide the underlying complexity from the end user.

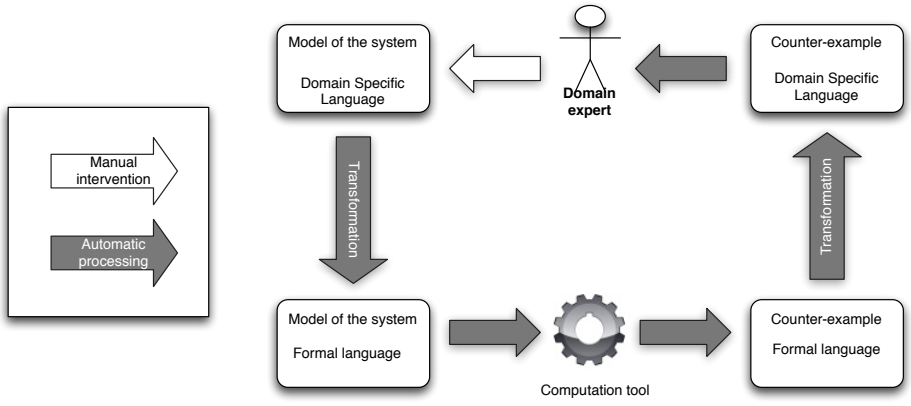


Fig. 3: DSL computational process

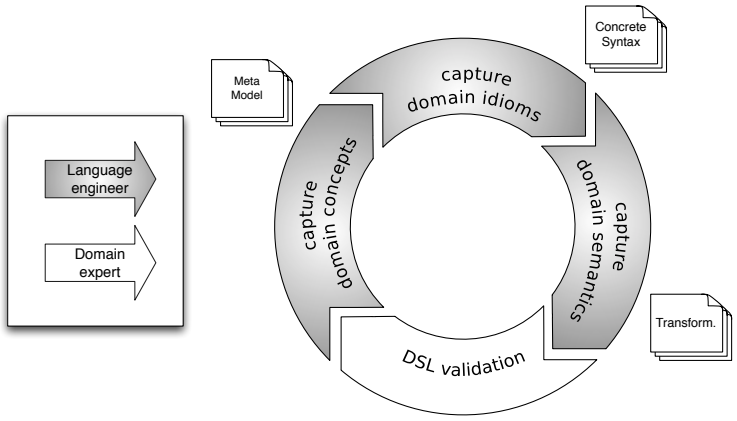


Fig. 4: DSL creation process

The person in charge of the DSL creation is a language engineer. This person should obviously have a certain knowledge about the language creation process, but he should also master the target platform language, in order to define efficient and correct transformations. Furthermore, he should be in contact with at least one domain expert, in order to settle the requirements and verify the correctness and completeness of the language created. The creation of a DSL follows a set of specific steps. First, the language engineer must identify the abstract concepts of the domain. These concepts include the basic elements of the domain, the interactions between these elements and the precise boundaries of the domain considered. Based on these concepts, the language engineer must define a set of expressions used to create a specific model in the domain, *i.e.,* a concrete syntax. Finally, the language engineer must define the semantics of the language created, usually by transformation to an existing platform. The whole process must be validated by one or more domain experts. Domain experts must validate the three steps of the DSL creation process: the domain must have been correctly defined, the expressions of the concrete syntax must be close to the already existing languages in the domain, and the execution must return the expected results. This creation/validation process often leads to an iterative development of the language. We show this entire process in Figure 4.

There exist various DSLs tailored for the biological processes, *e.g.,* SBML [8] and SBGN [11]. These two well-known languages cover a wide range of systems, mainly in the bio-chemical domain. GReg, instead, focuses on a more specific domain, which is genetic regulatory mechanisms. This domain has been described in Section 2.

While creating GReg, we used the Eclipse Modeling Project (EMP)[1] approach. We first created a metamodel of the domain using the Eclipse Modeling Framework (EMF), and we defined a concrete syntax with XText. XText provides a set of tools to create an editor for a given language, with some user-friendly features such as syntax highlighting, on the fly syntax checking (see Figure 5) and auto-completion. As a target platform we chose AlPiNA. AlPiNA[3]

is a model checking tool for Algebraic Petri Nets (APNs). It aims to perform efficient model checking on models with extremely large state spaces, using Decision Diagrams (DDs) to tackle the state explosion problem. AlPiNA's input languages were also defined using the EMP approach. This allowed us to use Atlas Transformation Language (ATL) transformations, which is a tool dedicated to define model to model transformations. GReg is thus fully integrated in the Eclipse/EMP framework.

The modular structure of GReg's definition would allow us to replace the target domain while keeping exactly the same language. If needed, we could, for example, define a transformation to SBML using a model to text transformation tool like XPand.

5 GReg : Gene Regulation Language

GReg is a Domain Specific Language designed to describe genetic regulatory mechanisms. We built it in order to illustrate the DSL approach, and the benefits it provides to research in the life sciences domain. Throughout this section, we introduce the GReg language using an excerpt of the lac operon model [10]. We also introduce the GReg Query Language (GQL) language, used to specify the queries to be executed in the model specified in GReg.

We first show how to describe a regulation mechanism. Listing 1 shows the overall structure of a GReg mechanism specification. The mechanism is named (`lac_operon`). It specifies the **molecules** occurring in the mechanism, and the chemical **reactions** between these molecules. The GReg description also specifies the **genes** with their properties and organization into **chromosomes**.

```
mechanism lac_operon is
  molecules
  -- declaration of molecules
  reactions
  -- declaration of chemical reactions
  chromosomes
  -- declaration of chromosomes
  gene -- declaration of a gene
  -- declaration of other genes
end lac_operon
```

Listing 1: GReg mechanism specification

The **molecules** section of a GReg description specifies the molecules occurring in the mechanism. For instance, in Listing 2, the `lac_operon` mechanism uses molecules `lactose`, `allolactose`, `lacI`, `lacZ`, *etc.*

```
mechanism lac_operon is
  molecules
    lactose, allolactose,
    lacI, lacZ, lacY, lacA,
    cAMP, CAP
  ...
end lac_operon
```

Listing 2: Molecules declaration

Molecules are only described by their names, as it is the only information relevant in our language. The DSL approach emphasizes specification of

only the required information for the particular domain. No molecules other than the ones described here can be used in the mechanism. This constraint is useful for the user creating a GReg specification: spelling errors in molecule names are detected, see Figure 5.

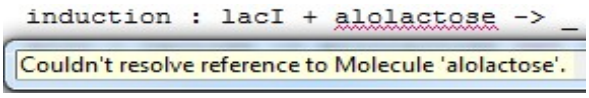


Fig. 5: Example of a spelling error in a molecule name (`allolactose`).

After the molecules declaration, a GReg description specifies the chemical reactions that take part in the mechanism. Listing 3 presents the **reactions** part of the mechanism.

```
mechanism lac_operon is
  ...
  reactions
    induction : lacI + allolactose  →  _
    allo      : lactose  →  allolactose cat lacZ
  ...
end lac_operon
```

Listing 3: Reactions declaration

In GReg, each reaction has a name, for instance `induction`. As usual in chemical notations, a reaction is a relation among (weighted) molecules. The molecule weight is usually called the stoichiometric coefficient. By default, stoichiometric coefficients are valued `1`.

A reaction can be either irreversible ($\rightarrow$) or reversible ($\leftrightarrow$). In our example, the reaction `induction` is a degradation of `lacI` and `allolactose`. A degradation is an irreversible reaction where products are not interesting, thus the reactants are simply removed from the system.

If needed, each direction of the reaction can be given a reaction rate (*i.e.,* probability). Each reaction can also have catalysts specified using the keyword **cat**. For instance `allo` is a reaction catalyzed by `lacZ`.

Listing 4 presents the **gene**s specification. For instance, `rep` is a minimal gene (*i.e.,* not regulated). A minimal gene defines at least the molecules it **codes**. If they are relevant, regulation sites are also specified in a section with the **sites** keyword. The `lac` gene defines a regulated gene with two regulation sites **I** and **O** , together with the molecule acting on them. Note that GReg also allows us to define several regulation sites for **I**, **O**, **A**, **T**.

```
mechanism lac_operon is
  ...
  gene rep
    codes lacI
  end rep
  gene lac
    codes lacZ, lacY, lacA
    sites
      I : cAMP and CAP = 1
      O : lacI @ 2
  end lac
end lac_operon
```

Listing 4: Genes declaration

As molecules may be present at different levels, the @ keyword allows the specification of the required molecules levels acting at a regulation site. By default, molecule levels are valued 1. As several molecules can act on one regulation site, GReg allows to combine molecules with Boolean operators for a regulation site. There are two operators defined : **and** and **or**. For instance the **I** site of `lac` gene specifies that `cAMP` **and** `CAP` are required to activate this site. The = keyword allows to specify the target level attributed to the gene once this site is active.

Note that for A sites, it is also possible to specify the next gene target level. As the role of a T site is to interrupt the transcription process, we allow the specification of the reduced set of produced molecules when these sites are active.

The **chromosomes** section is used to specify one or more chromosomes. A chromosome defines the sequence of loci. A locus is defined between two braces. Note that genes'order in each locus does not matter. This section is mandatory when taking into account A sites.

```
mechanism lac_operon is
  ...
  chromosomes
    c : {rep}, {lac,lac'}
  ...
end lac_operon
```

Listing 5: Chromosomes declaration

Listing 6 shows an example of GQL specification. The **use** keyword imports the `lac_operon` mechanism from another file, allowing us to reference the molecules declared in `lac_operon` mechanism from a query specification. A GQL file specifies the **levels** and the **queries**.

The **levels** section is used to define the combination of levels. A combination of levels is a partial or total definition of molecule levels, while unspecified molecules may match any possible value. For instance `l1` specifies only the level of `lacZ` among all molecules defined in `lac_operon`. The **exists** query returns true if predefined level exists. The **paths** query is used to retrieve all paths from the state space matching the sequence of predefined levels. The query `b` returns the path where `l2` is a direct successor of `l1`. But query `c` does not require that `l2` is a direct successor of `l1`.

```
use "lac_operon.greg"
levels
  l1 : lacZ = 1
  l2 : lacZ = 0, lacI = 2
queries
  bool a  : exists l1
  paths b : paths l1, l2
  paths c : paths l1 .. l2
```

Listing 6: GQL queries specification

Listing 7 shows an example of a GReg configuration specification given outside the mechanism, usually in a separate file. This allows to easily repeat experiments for the same mechanism with several initial quantities. The first section starts with the **initially** keyword and defines the genes or molecules initial levels. The second section starts with the **execute**

```
use "lac_operon.greg"
use "lac_operon.gql"
initially lac_operon has
  lactose = 1
  lacI = 1
execute
  if a then (b and c)
```

Listing 7: GReg specification

keyword and is used to specify which queries will be executed by the model checker.

6 Example

We present a simple example of a genetic regulatory mechanism taken from [17] with three genes. Gene Y is activated by the product of gene X. Genes X and Z are repressed by the products of genes Z and Y respectively. The products of genes X, Y and Z are molecules x, y and z respectively. Graphical (LRG) and textual (GReg) models derived from this example are given at Figure 6 and Listing 8 respectively.

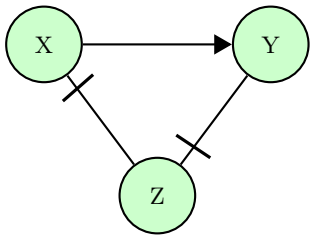


Fig. 6: LRG model of example

```
mechanism example is
  molecules x, y, z
  gene X codes x
    sites O : z
  end X
  gene Y codes y
    sites I : x
  end Y
  gene Z codes z
    sites O : y
  end Z
end example
```

Listing 8: GReg model of Figure 6

GReg models are transformed into APN models usable by AlPiNA. Note that the obtained APNs can always be unfolded into Place/Transition Petri Nets (P/Ts). We propose two different transformations:

- the first transformation produces a P/T shown in Figure 7, called Multi-level Regulatory Petri net (MRPN) in [5] ;
- the second transformation produces an APN shown in Figure 8, which is the folding of the corresponding MRPN and thus more compact.

From theses models AlPiNA is able to compute the state space and to identify all deadlock without requiring any additional input from the user.

A *deadlock* is a situation where no more events can occur in the system. Strictly speaking, in real biological systems there are no deadlocks but *livelocks*, a situation where events still occur but without changing the state of the system. The states where such situations occur are usually called stable states or attractors.

But we can also search for specific properties of the biological system. As previously mentioned, the state space might contain too many states to be used as it is. Therefore we propose a way to extract portions of state space (*e.g.*, subsets

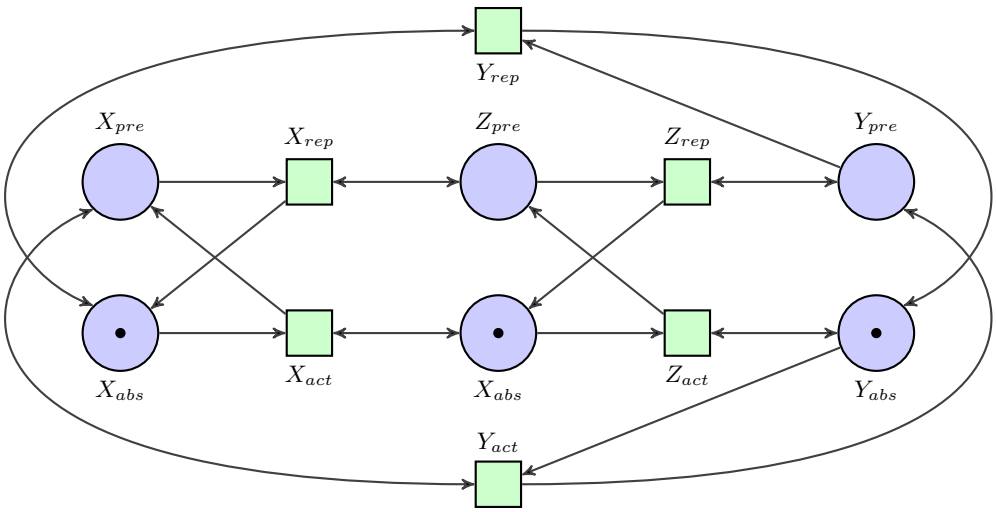


Fig. 7: PN model of example in Figure 6 and Listing 8

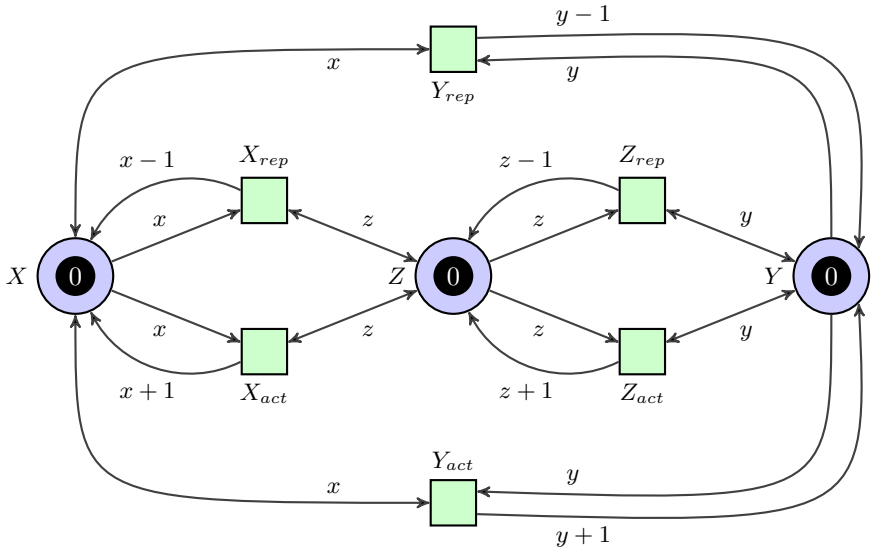


Fig. 8: APN model of example in Figure 6 and Listing 8

of states, paths, cycles, ...) through GQL queries. In Listing 9, we have defined one level (`l1`) and one query (**at**).

```
use "example.greg"
levels
  l1 : x = 1, y = 1
queries
  states s1 : at l1
```

Listing 9: GReg query example

This query returns the subset (`s1`) of states from the state space where the level of `x` and `y` is equal to one. To compute the set of states `s1`, the query is transformed into an AlPiNA property, shown in Listing 10.

```
s1 : exists($x in x, $y in y, $z in z :
 (($x equals suc(zero)) and ($y equals suc(zero))) = false
```

Listing 10: AlPiNA property expression of query in Listing 9

The example in Figure 6 and Listing 8 has eight states reachable from the initial marking, where all molecules are initially at level zero (`x,y,z`) = (0,0,0), see Figure 9. Model checking of the example PN finds two stable states: (1,1,0) and (0,0,1) and returns for `s1` the two states: (1,1,0) and (1,1,1).

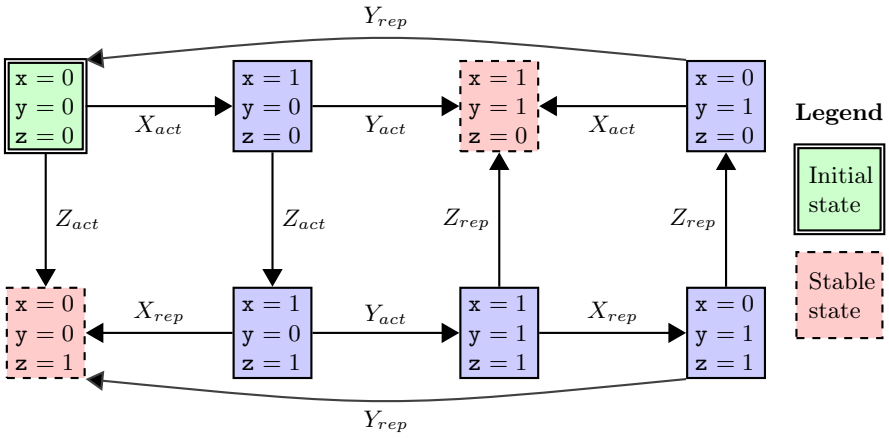


Fig. 9: State space of example in Figure 6 and Listing 8

7 Conclusion & future work

This paper introduces Gene Regulation Language (GReg), a language dedicated to the modeling of regulatory mechanisms. We explain the need to explore completely the sets of possible behaviors of a given model in order to detect rare events. GReg includes a query language used to express the properties of such events.

From a technical point of view, we explain that the techniques proposed are based on general principles borrowed from software modeling and verification. These techniques include the use of a dedicated DSL defined with a metamodeling approach and the translation of this language into a formal verification platform called AlPiNA.

The languages and transformations shown in this article have been implemented and tested on several toy examples. We also asked biologists to assess the expressivity and usability of the language. Although the first feedback seems promising, there is much room for improvement. We foresee three main axes of future development: improving the expressivity of the modeling and query languages, assessing the usability of the approach and exploring the mitigation of the state space explosion.

Extending the expressivity of GReg Concepts such as time and probabilities play an important role in biology and are therefore good candidates for a language extension. As the current underlying formalism, APNs, does not support these notions, such extension would require changing the target platform. Good examples of target formalisms are timed Petri nets and stochastic Petri nets. As mentioned in 4, the techniques used to create GReg allow changing the target language without changing the language itself. Note that we do not plan to add continuous concepts, used in languages such as HFPNe.

Improving the usability of our tool Textual domain specific languages constitute a first step towards democratization of formal methods. Although highly efficient, textual languages are usually not as intuitive as graphical languages. On the other hand, graphical domain specific languages are especially good in the early phase of the modeling as well as for documentation, but they are often less practical when the model grows. The tools in EMP that were used to create GReg allow us to define a graphical version of the same language, thus keeping the best of both worlds.

Another way to ease the modeling phase is to allow import/export of models from/to other formalisms and standards such as SBML and to integrate it with *Cytoscape* through its plug-in mechanism.

Mitigating the state space explosion So far, we have done little experimentation in this area for biological processes. Nevertheless, we conducted several studies on usual IT protocols and software models that show that AlPiNA can handle huge state spaces[2]. This suggests promising results in the regulatory mechanisms domain.

The development of GReg is a work in progress, we would like to set up more collaborations with biologists interested in exploiting formal techniques from computer science to discover rare events. We think that we can make, in the near future, a useful contribution to life sciences based on advanced techniques borrowed from computer science.

References

1. Eclipse Modeling Project. `http://www.eclipse.org/modeling/`.
2. D. Buchs, S. Hostettler, A. Marechal, and M. Risoldi. AlPiNA: A Symbolic Model Checker. In *Petri Nets'2010: Applications and Theory of Petri Nets, Braga, Portugal*, volume 6128 of *Lecture Notes in Computer Science*, pages 287–296, 2010.
3. D. Buchs, S. Hostettler, A. Marechal, and M. Risoldi. AlPiNA: An Algebraic Petri Net Analyzer. In *TACAS'2010: Tools and Algorithms for the Construction and Analysis of Systems, Paphos, Cyprus*, volume 6015 of *Lecture Notes in Computer Science*, pages 349–352, 2010.
4. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation*, 98, 1992.
5. C. Chaouiya, H. Klaudel, and F. Pommereau. A Modular, Qualitative Modeling of Regulatory Networks Using Petri Nets. In *Modeling in Systems Biology*, volume 16 of *Computational Biology*, pages 253–279. Springer London, 2011.
6. C. Chaouiya, E. Remy, B. Mossé, and D. Thieffry. Qualitative Analysis of Regulatory Graphs: A Computational Tool Based on a Discrete Formal Framework. In *Positive Systems*, volume 294 of *Lecture Notes in Control and Information Sciences*, pages 830–832. 2003.
7. Cytoscape Consortium and Funding Agencies. `http://www.cytoscape.org/`.
8. M. Hucka, F. Bergmann, S. Hoops, S. Keating, S. Sahle, J. Schaff, L. Smith, and B. Wilkinson. The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core, 2010. Available from Nature Precedings `http://dx.doi.org/10.1038/npre.2010.4959.1`.
9. J-M. Couvreur and E. Encrenaz and E. Paviot-Adet and D. Poitrenaud and P-A. Wacrenier. Data Decision Diagrams for Petri Net Analysis. In *ATPN'02: International Conference on Application and Theory of Petri Nets*, volume 2360 of *Lecture Notes in Computer Science*, pages 101–120, 2002.
10. F. Jacob and J. Monod. Genetic Regulatory Mechanisms in the Synthesis of Proteins. *Journal of molecular biology*, 3:318–356, 1961.
11. S. Moodie, N. Le Novere, E. Demir, H. Mi, and A. Villeger. Systems Biology Graphical Notation: Process Description language Level 1, 2011. Available from Nature Precedings `http://dx.doi.org/10.1038/npre.2011.3721.4`.
12. M. Nagasaki, A. Doi, H. Matsuno, and S. Miyano. A Versatile Petri Net Based Architecture for Modeling and Simulation of Complex Biological Processes. *Genome Informatics*, 15(1):180–197, 2004.
13. A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya. Logical Modelling of Regulatory Networks with GINsim 2.3. *Biosystems*, 97(2), 2009.
14. A. Naldi, C. Chaouiya, and D. Thieffry. GINsim. `http://gin.univ-mrs.fr/`.
15. P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Research*, 13, 2003.
16. The University of Tokyo. CellIllustrator. `http://www.cellillustrator.com/`.
17. R. Thomas. Regulatory Networks seen as Asynchronous Automata: a Logical Description. *Journal of Theoretical Biology*, 153:1–23, 1991.
18. A. Valmari. The State Explosion Problem. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, pages 429–528, 1998.

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.36-52

Modelling Reaction Systems with Petri Nets (Extended Abstract)

Jetty Kleijn[1], Maciej Koutny[2], and Grzegorz Rozenberg[1,3]

[1] LIACS, Leiden University, 2300 RA, The Netherlands
[2] School of Computing Science, Newcastle University, NE1 7RU, UK
[3] Department of Computer Science, University of Colorado at Boulder
430 UCB Boulder, CO 80309-0430, U.S.A.

Abstract. We investigate how Petri nets could be used to provide a faithful semantics of reaction systems, a formal framework for the investigation of processes carried by biochemical reactions. We propose and discuss possible approaches to this problem using some existing Petri net classes and concurrency concepts, such as maximal parallelism. After that we introduce a new class of Petri nets, called SET-nets, which provide a computational model matching very closely that exhibited by reaction systems. The key difference between standard Petri nets and SET-nets is that the former support multiset-based token arithmetic, whereas the latter support set-based operations on tokens.
Keywords: reaction system, Petri net, living cell, natural computing, SET-net, model translation

1 Introduction

The investigation of the computational nature of biochemical reactions is a research topic of Natural Computing. One of the goals of this research is to contribute to a computational understanding of the functioning of the living cell.

Reaction systems [2, 3, 7–10] are a formal framework for the investigation of processes carried out by biochemical reactions in living cells. The central idea of this framework is that the functioning of a living cell is based on interactions between (a large number of) individual reactions, and moreover these interactions are regulated by two main mechanisms: facilitation/acceleration and inhibition/retardation. These interactions determine the dynamic processes taking place in living cells, and reaction systems form a formal framework for developing an abstract theory of these processes.

The model of reaction systems is based on principles remarkably different from those underlying other *existing models of computation*. The aim of this paper is to develop a faithful Petri net model of reaction systems. The main motivation behind this is to establish whether Petri net based concepts (such as causal processes) and methods (such as synthesis of nets from a specification of their behaviour) could be used to provide analytical tools for reaction systems. It is not the intention of this paper to provide direct feedback to the area of

biological applications, but to establish bridges between biology and Petri nets through the connection provided by reaction systems.

As a first step, we propose and discuss four different approaches to the modeling of reaction systems by using existing Petri net classes and concurrency concepts. However, as it turns out, in order to obtain a good match between reaction systems and Petri nets, it is necessary to re-evaluate one of the basic net principles, namely, token counting. This leads us to the introduction of a new class of Petri nets, called SET-nets, which provide a net based computational model matching very closely the computations exhibited by reaction systems. The main difference between SET-nets and standard Petri nets is that the latter support multiset-based token arithmetic, whereas the former support set-based (boolean) operations on tokens. Thus, the computational 'intuition' originating from reaction systems provides the inspiration to introduce a new class of nets with intriguing and yet to discover properties. Consequently, the main contribution of this paper is more than just providing a bridge between reaction systems and the world of Petri nets. In the future, after fully understanding and mastering the properties of the new SET-nets, one would hope to provide also a new set of tools and analyses for biological applications.

The paper is organised in the following way. In the next section, we describe basic notions of reaction systems. Section 3 describes two methods of modelling reaction system using low-level Petri nets, and the next one does the same using high-level Petri nets. The new class of SET-nets is introduced in Section 5, and in Section 6 we explain why this new class of nets can faithfully and elegantly model reaction systems. Comparison with related work is presented in Section 7. Proofs of the results presented in this paper can be found in [16].

Notation We use the standard mathematical notions and notation. A multiset over a set X is a function $\mu : X \to \mathbb{N} = \{0, 1, 2, \ldots\}$, and its support is $\|\mu\| = \{x \in X \mid \mu(x) > 0\}$. The empty multiset $\varnothing$ satisfies $\|\varnothing\| = \varnothing$. A multiset may be represented, somewhat informally, by listing its elements with repetitions, e.g., $\mu = \{y, y, z\}$ is such that $\mu(y) = 2$, $\mu(z) = 1$, and $\mu(x) = 0$ otherwise. We treat sets as multisets without repetitions.

2 Reaction systems

In this section, we explain some notions relevant to reaction systems. It is our intention to introduce enough concepts to allow one to follow the subsequent discussion on the relationship between reaction systems and Petri nets. For a comprehensive description of reaction systems, including motivations, applications and examples, the reader is referred to [7–9].

Definition 1 (reaction system [7–9]). *A* reaction system *is a pair:* $\mathcal{A} = (S, A)$, *where* S *is a finite* background *set comprising the* entities *of* $\mathcal{A}$, *and* A *is the set of reactions of* $\mathcal{A}$. *Each* reaction *is a triplet of the form:* $a = (R, I, P)$,

where the three components are finite non-empty sets: $R \subseteq S$ *is the set of* reactants, $I \subseteq S$ *is the set of* inhibitors, *and* $P \subseteq S$ *is the set of* products.

The components of a reaction $a = (R, I, P)$ are denoted by R_a, I_a and P_a, respectively. Definition 1 describes the *static* structure of a reaction system. To capture the *dynamic* behaviour of reaction systems, we need additional notions.

Definition 2 (state of reaction system). *A* state *of a reaction system is any set* C *of its entities. Then an* initialised *reaction system is a triplet* $\mathcal{A} = (S, A, C_0)$, *where* (S, A) *is a reaction system and* $C_0 \subseteq S$ *is the* initial *state.*

In this and in the next section, we will consider as a running example the initialised reaction system $\mathcal{A}_0 = (\{w, x, y, z\}, \{a, b, c\}, \{x, z\})$, with background set $\{w, x, y, z\}$, initial state $\{x, z\}$, and three reactions:

$$a = (\{x\}, \{y\}, \{y, z\}) \quad b = (\{y\}, \{x\}, \{x, z\}) \quad c = (\{z\}, \{w\}, \{z\}) \,.$$

A reaction system with background set S has exactly $2^{|S|}$ potential states. To describe possible transitions between these states, we need to say what is meant by an occurrence of a reaction or a set of reactions.

Definition 3 (state change). *A reaction* a *is* enabled *at a state* $C \subseteq S$ *if* $R_a \subseteq C$ *and* $I_a \cap C = \varnothing$; *the* result *of a reaction* a *at* C *is defined by* $res_a(C) = P_a$ *if* a *is enabled at* C *and* $res_a(C) = \varnothing$ *otherwise. The result of* A *on* C, *denoted by* $res_{\mathcal{A}}(C)$ *consists of the products of all reactions from* A *enabled at* C, *that is*

$$res_{\mathcal{A}}(C) = \bigcup_{a \in A} res_a(C) \,.$$

This state change is denoted by $C \longrightarrow res_{\mathcal{A}}(C)$.

Note that the state changes captured by Definition 3 are deterministic. Moreover, all entities in $C \setminus \bigcup_{a \in A} res_a(C)$ disappear. As a result, and unlike in other formal models of dynamic systems, there is no persistency in a reaction system in the sense that an entity present in a state disappears unless it is sustained by at least one reaction.

For the example reaction system $\mathcal{A}_0$, we have:

$$\{x, z\} \longrightarrow \{y, z\} \;\text{ and }\; \{y, z\} \longrightarrow \{x, z\} \;\text{ and }\; \{w, x, y\} \longrightarrow \varnothing \,.$$

One may observe that there is no conflict between reactions in the 'classic' sense that the occurrence of one reaction might imply that another reaction which is also enabled at the current state, cannot occur. This, again, is a feature not found in most other formal models of dynamic systems. In particular, it is worthwhile to point explicitly to the 'non-counting' features of reaction systems: entities are either present or not, and produced or not, and reactions can or cannot occur based only on the presence or absence of certain entities. There is no representation of multiple instances of entities or multiple occurrences of

reactions. Thus reaction systems are a *qualitative* rather than a quantitative model.

We also note that there is an alternative notion of conflict-freeness for a set of reactions, called consistency. A set of reactions $\mathcal{R}$ is *consistent* if for any two reactions $a, b \in \mathcal{R}$, $R_a \cap I_b = R_b \cap I_a = \varnothing$. Clearly, if a set of reactions is *not* consistent, then the reactions it comprises cannot be executed simultaneously.

Although the goal of this paper is a faithful 'translation' of reaction systems into Petri nets, we conclude this section with a number of comments about research on reaction systems. This research happens in the *framework* of reaction systems where a reaction system constitutes the basic technical notion. Depending on the goal of a specific research theme, many other constructs are introduced and studied (see, e.g., [2, 9, 10]) — they form various extensions of the basic notion of reaction system. For example, there are many biological situations where one needs to assign quantitative parameters (time, concentrations, ...) to states of a biochemical system. Although reaction systems are a qualitative model (they cannot 'count'), they can be extended so that such quantitative parameters can be accommodated. This is done through the use of *measurement functions* which lead to *reaction systems with measurements* (see [2, 3, 9, 10]), where various numerical parameters can be assigned to (calculated for) consecutive states of dynamic processes.

Finally, we want to point out that (because living cells are open systems) reaction systems have an environment and they operate/evolve within a changing context (with entities coming from the environment influencing the transitions of dynamic processes). In this paper, however, we will consider only *context-independent* processes defined by a reaction system with an initial state, where each next state is obtained solely as the result of reactions taking place in the previous state (thus assuming that the environment does not influence state transitions).

3 Reaction systems and low-level Petri nets

In this section, we discuss two possible ways of modelling context-independent processes of reaction systems using low-level Petri nets (PT-nets extended with with inhibitor and activator arcs).

In addition to the standard notions of reaction systems, in order to better explain how they relate to Petri nets, throughout the rest of this paper we will say that a set $\mathcal{R} \subseteq A$ is enabled at C if each reaction of $\mathcal{R}$ is enabled at C. If $\mathcal{R} \subseteq A$ is enabled at C, then

$$C \stackrel{\mathcal{R}}{\Longrightarrow} res_{\mathcal{R}}(C) = \bigcup_{a \in \mathcal{R}} P_a \ .$$

denotes the effect of $\mathcal{R}$ at C.

Definition 4 (PT-nets with inhibitor and activator arcs [14]). *A* PT-net with inhibitor and activator arcs *(or* PTIA-*net)* $N = (Pl, Tr, Flw, Inh, Act, M_0)$

is a tuple such that Pl and Tr are finite, disjoint sets of respectively places *and* transitions*, and: Flw* $\subseteq (Pl \times Tr) \cup (Tr \times Pl)$*, Inh* $\subseteq Pl \times Tr$*, Act* $\subseteq Pl \times Tr$ *are respectively the sets of* flow*,* inhibitor *and* activator *arcs. Moreover,* M_0 *is a multiset of places, the* initial *marking of* N*; in general, any multiset of places is called a* marking.

In diagrams, places are drawn as circles and transitions as rectangles. Markings are the possible global configurations (states) of N. We say that a place q is *marked* under a marking M if $M(q) > 0$, where $M(q)$ denotes the number of occurrences of q in M. In diagrams, markings are indicated by putting $M(q)$ *tokens* inside the circle representing q. If $(x, y) \in Flw$, then (x, y) is an *arc* leading from *node* x to *node* y. A double headed arrow between q and t indicates that $(q, t), (t, q) \in Flw$. An inhibitor arc ends with a small open circle, while an activator arc ends with a small black circle.

Given a node x, we denote by ${}^\bullet x$ the set of *input nodes* of x, i.e., those y for which $(y, x) \in Flw$, and by $x^\bullet$ the set of *output nodes* of x, i.e., those y for which $(x, y) \in Flw$. For a transition t we use: ${}^\circ t = \{q \mid (q, t) \in Inh\}$ and ${}^\blacklozenge t = \{q \mid (q, t) \in Act\}$ to denote the inhibitor and activator places of t. All four notations extend in the usual way to sets of nodes. As in the case of reaction systems, we now formalise the notion of marking (state) change.

Definition 5 (marking change). *A multiset of transitions* U *(also called a* step*) is* enabled *at a marking* M *if* ${}^\circ U \cap \|M\| = \varnothing$*,* ${}^\blacklozenge U \subseteq \|M\|$ *and, for every place* q*,* $M(q) \geq \sum_{t \in q^\bullet} U(t)$ *(recall that* $\|M\|$ *is the set of* q *which occur in* M*, and* $U(t)$ *is the number of occurrences of* t *in* U*).*
In such a case, U *can be* fired *with its effect on* M *being given by the resulting marking* M' *such that, for every place* q*:* $M'(q) = M(q) - \sum_{t \in q^\bullet} U(t) + \sum_{t \in {}^\bullet q} U(t)$*. We denote this by* $M[U\rangle M'$*. Moreover, if* U *is a* <u>*maximal*</u> *(w.r.t. multiset inclusion) step of transitions enabled at* M*, then we may denote this* marking change *also by* $M[U\rangle_{max} M'$*.*

Note that whenever a step U is enabled at marking M it must be the case that all activator places of transitions in $\|U\|$ are marked (are in $\|M\|$) and none of the inhibitor places of transitions in $\|U\|$ are marked.

We now make some general observations and assumptions about the relationship between reaction systems and nets.

- Entities can be represented by places, and reactions by net transitions.
- Since there are no conflicts between reactions, activator arcs can be used to test for the presence of reactants (rather than claiming resources for the exclusive use as with ordinary arcs and input places).
- All reactions that can occur in a reaction system do occur, and the only entities left after a state change are the newly generated products. In the Petri net framework, these features correspond to *maximal parallelism* described at the end of Definition 5, and *place resetting* [6] described later on.

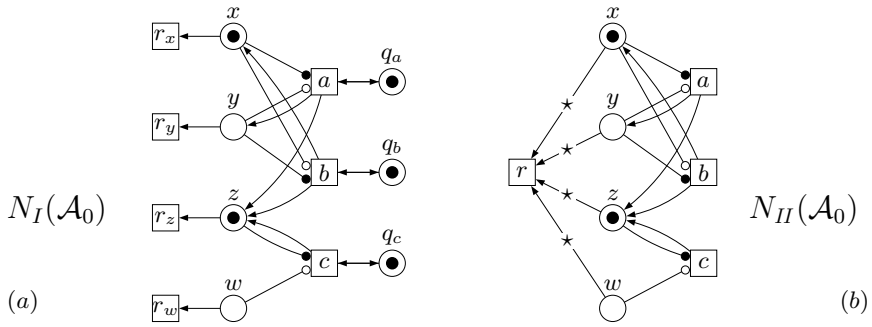


Fig. 1. Method I and II representations of the reaction system $\mathcal{A}_0$.

Method I. The first attempt is illustrated in Figure 1(a) for the example reaction system $\mathcal{A}_0$. Method I produces a PTIA-net $N_I(\mathcal{A}_0)$ such that:

- Transitions a, b and c use activator arcs and inhibitor arcs to test respectively for the presence and absence of tokens in the places w, x, y and z.
- Places q_a, q_b and q_c ensure that the three transitions modelling reactions, i.e., a, b and c, fire at most once in any step. This corresponds to the 'non-counting' of occurrence instances of the same reaction in a reaction system.
- Transitions r_w, r_x, r_y and r_z (in a maximal step) empty the four places modelling entities w, x, y and z. This does not have any influence on the firing of the transitions a, b and c.
- In a single maximal step, $M[U\rangle_{max}M'$, the net fires a maximal multiset of transitions U enabled at marking M and then produces a new marking M'. For the net in Figure 1(a), such a firing rule gives:

$$\{x,z,q_a,q_b,q_c\}\ [\{r_x,r_z,a,c\}\rangle_{max}\ \{y,z,z,q_a,q_b,q_c\}$$
$$\{x,x,x,z,q_a,q_b,q_c\}\ [\{r_x,r_x,r_x,r_z,a,c\}\rangle_{max}\ \{y,z,z,q_a,q_b,q_c\}\ .$$

Formally, given an initialised reaction system $\mathcal{A}=(S,A)$, Method I yields a PTIA-net $N_I(\mathcal{A})$ such that the places, transitions and the initial marking are, respectively: $Pl=\{q_a \mid a\in A\}\cup S$, $Tr=\{r_s \mid s\in S\}\cup A$ and $M_0=\{q_a \mid a\in A\}+C_0$. Moreover, the sets of flow, inhibitor and activator arcs are, respectively:

$$Flw=\{(s,r_s) \mid s\in S\}\cup\{(a,q_a),(q_a,a) \mid a\in A\}\cup\{(a,s) \mid a\in A\wedge s\in P_a\}$$
$$Inh=\{(s,a) \mid a\in A\wedge s\in I_a\} \qquad Act=\{(s,a) \mid a\in A\wedge s\in R_a\}\ .$$

Note that this kind of modelling in combination with the 'resetting' of places w, x, y and z in each fired step, implemented by the auxiliary transitions r_w, r_x, r_y and r_z, means that the resulting Petri net is bounded (in every reachable marking the multiplicity of each place is never more than the number of reactions of $\mathcal{A}$ if $\mathcal{A}$ has at least one reaction).

In order to relate the behaviour of the original reaction system $\mathcal{A}$ and its PTIA-net representation $N_I(\mathcal{A})$ just introduced, we need two mappings. The first

one takes a marking M of $N_I(\mathcal{A})$ and returns a state of $\mathcal{A}$, and the other takes a step U of transitions of $N_I(\mathcal{A})$ and returns a set of reactions of $\mathcal{A}$, as follows $\nu_I(M) = S \cap \|M\|$ and $\varphi_I(U) = A \cap \|U\|$. It is then possible to show a number of results, where a marking M of the PTIA-net $N_I(\mathcal{A})$ is called *well-formed* if $M(q_a) = 1$, for every $a \in A$.

First, M_0 is a well-formed marking satisfying $\nu(M_0) = C_0$, and if M is a well-formed marking and $M[U\rangle M'$, then M' is also well-formed. Second, if M is a well-formed marking, then for every reaction $a \in A$, a is enabled at M iff $\{a\}$ is enabled at state $\nu_I(M)$. We then can show that the translation is sound.

Theorem 1. *If M is a well-formed marking then:*

1. *$M[U\rangle M'$ implies $\nu_I(M) \stackrel{\varphi_I(U)}{\Longrightarrow} \nu_I(M')$. Moreover, if $M[U\rangle_{max} M'$, then $\varphi_I(U)$ comprises all reactions enabled at $\nu_I(M)$.*
2. *$\nu_I(M) \stackrel{\mathcal{R}}{\Longrightarrow} C$ implies $M[U\rangle M'$ for some U and M' satisfying: $\varphi_I(U) = \mathcal{R}$ and $\nu_I(M') = C$. Moreover, if $\mathcal{R}$ comprises all reactions enabled at $\nu_I(M)$, then $M[U\rangle_{max} M'$.*

Thus, each maximal computational step in the Petri net corresponds to a unique execution of the reaction system, and each execution in the reaction system corresponds to at least one maximal step in the Petri net. For example, the two executions given above for the Petri net in Figure 1(a) both correspond to $\{x, z\} \stackrel{\{a,c\}}{\Longrightarrow} \{y, z\}$ in the reaction system $\mathcal{A}_0$.

Note that in Figure 1(a) one cannot simply delete the auxiliary places of the form q_r as then each of the transitions representing reactions could be unboundedly enabled. To address this problem one could change the activator arcs from places representing entities into flow arcs. Then, however, it would be necessary to add weights $|R|$ to the arcs corresponding to the production of new entities in order to avoid conflicts on the places representing the reactants.

Method II. The first attempt to model context-independent reaction systems provides a sound translation, but it is not simple as it employs features which can make formal analysis and verification far from easy. One way of improving Method I could be to replace multisets of fired transitions by sets of fired transitions leading to a *maximal set-semantics*. This can be achieved by using *reset* arcs [6], connecting places to transitions and indicated by $\star$'s in the diagrams, which always empty their source place. Formally, reset arcs $Reset \subseteq Pl \times Tr$ do not have any influence on the enabledness of a step U, but the calculation of the marking of a place q after the firing of U (now a set) at marking M changes to:

$$M'(q) = \begin{cases} M(q) - |q^\bullet \cap U| + |{}^\bullet q \cap U| & \text{if } (\{q\} \times U) \cap Reset = \varnothing \\ |{}^\bullet q \cap U| & \text{otherwise .} \end{cases}$$

The resulting PTIA-net with reset arcs $N_{II}(\mathcal{A}_0)$ is shown in Figure 1(b). Transition r is always enabled and, when fired, removes all the tokens from the places modelling the entities. For the net in Figure 1(b), the new firing rule gives

$\{x,z\}\ [\{r,a,c\}\rangle_{max}\ \{y,z,z\}$ and $\{x,x,x,z\}\ [\{r,a,c\}\rangle_{max}\ \{y,z,z\}$. One can then show that a counterpart of Theorem 1 holds also in this case, with ν_{II} defined as ν_I before and $\varphi_{II}(U) = U \setminus \{r\}$. As transition r is always enabled, we now have a one-to-one correspondence between groups of executed reactions and transitions, at the price of introducing non-standard reset arcs.

To remove the need to have reset arcs or, equivalently, to obtain a one-to-one correspondence between states and markings, one could change the rules for inserting tokens into places, by basically applying an OR-treatment for arriving tokens. This would, of course, be a radical departure from the standard Petri net approach, but one worth investigating. The resulting model of SET-*nets* will be described in Section 5.

4 Reaction systems and high-level Petri nets

The two translations described in the previous section use low-level PT-nets extended with reset arcs in addition to inhibitor and activator arcs as well as maximal parallelism. Reset arcs are a non-standard mechanism and, in particular, they do not as yet support a causal process semantics. Moreover, the effect of a reset arc depends on the current marking rather than on a fixed input/output relation with its neighbourhood. To cope with this problem, we will now outline two translations from context-independent reaction systems to high-level Petri nets. We assume familiarity with the basic concepts of high-level nets [13], in particular, arc inscriptions, activator and inhibitor arcs, and simple transition guards.

Method III. The first translation is illustrated by the high-level net $N_{III}(\mathcal{A}_0)$ shown in Figure 2(a). In this case, tokens are positive integers acting as though they were time-stamps. Intuitively, a token n is active only in the n-th execution cycle of the reaction system. Because the same token cannot be accessed more than once in a step sequence evolution, reset arcs are not needed anymore. Since the ✓ transition fires in each maximal step, the cycle number n held in the 'clock' place *clk* is known to all transitions representing reactions. In the places representing entities, they check only for tokens n, ignoring all the other tokens produced in previous cycles, and then produce tokens with value $n+1$ to be used in the next cycle. The initial marking M_0 is formed by inserting a single token 1 into place *clk* and all the places s such that $s \in C_0$. Note that the resulting net may be unbounded as the tokens in places representing entities are not 'garbage collected'. For the high-level net $N_{III}(\mathcal{A}_0)$ in Figure 2(b), we have:

$$\begin{gathered}\{x \mapsto \{1\}, y \mapsto \varnothing, z \mapsto \{1\}, w \mapsto \varnothing, clk \mapsto \{1\}\}\\ [\{a_{n\mapsto 1}, c_{n\mapsto 1}, \checkmark_{n\mapsto 1}\}\rangle_{max}\\ \{x \mapsto \{1\}, y \mapsto \{2\}, z \mapsto \{1,2,2\}, w \mapsto \varnothing, clk \mapsto \{2\}\}\\ [\{b_{n\mapsto 2}, c_{n\mapsto 2}, \checkmark_{n\mapsto 2}\}\rangle_{max}\\ \{x \mapsto \{1,3\}, y \mapsto \{2\}, z \mapsto \{1,2,2,3,3\}, w \mapsto \varnothing, clk \mapsto \{3\}\}\ .\end{gathered}$$

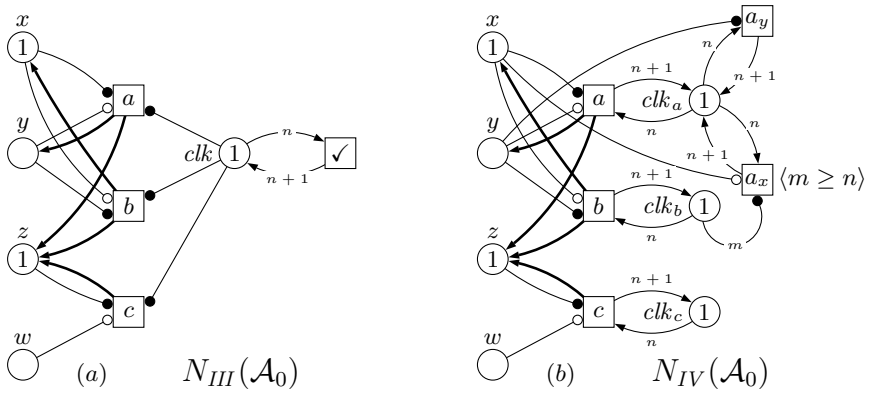


Fig. 2. Method III and IV representations of reaction system $\mathcal{A}_0$. Note that n and m are net variables, and that to avoid clutter not all arcs have been annotated: all the flow (thicker) arcs to places x, y, z are in fact annotated with $n+1$, and all the unannotated inhibitor and activator arcs are annotated with n. In (b), the auxiliary places for transitions b and c are omitted. Note that $\langle m \geq n \rangle$ is the guard of transition a_x, and all other transitions have the trivial *true* guard.

As in the case of Method I, not every marking M of $N_{III}(\mathcal{A})$ can represent a valid state of the reaction system $\mathcal{A}$. We say that M is *clock-consistent* if there is a single token k in place *clk*, and all the tokens l in other places satisfy $l \leq k$.

Relating the resulting net and the original reaction system can be done using the following two mappings: $\nu_{III}(M) = \{s \in S \mid \|M(clk)\| \cap \|M(s)\| \neq \varnothing\}$ and $\varphi_{III}(U) = U \setminus \{\checkmark\}$. One can show that M_0 is a clock-consistent marking satisfying $\nu(M_0) = C_0$, and if M is a clock-consistent marking and $M[U\rangle_{max} M'$ then M' is also clock-consistent.

Theorem 2. *If M is a clock-consistent marking then:*

1. *$M[U\rangle_{max} M'$ implies $\nu_{III}(M) \overset{\varphi_{III}(U)}{\Longrightarrow} \nu_{III}(M')$.*
2. *$\nu_{III}(M) \overset{\mathcal{R}}{\Longrightarrow} C$ implies $M[U\rangle_{max} M'$ for some U and M' satisfying: $\varphi_{III}(U) = \mathcal{R}$ and $\nu_{III}(M') = C$.*

Method IV. In the second high-level net construction the aim is to eliminate the need for maximal parallelism using information present in the time-stamped tokens. We replace the global *clk* place by individual clk_a places, which are incremented by transitions a representing reactions. Moreover, whenever a is blocked from firing in a certain cycle one of the auxiliary transitions corresponding to the possible 'reasons' for the blocking a is fired to increment the token in clk_a. This results in an increment of the cycle number for this transition (in case

there is more than one reason for blocking, an auxiliary transition is chosen non-deterministically).

There are two possible reasons why a might be blocked in cycle n. One is the presence of a token n in the place representing an inhibitor of a, and to check for this we use a transition with an activator arc, e.g., a_y in Figure 2(b). The other is more complicated as it is a lack of token n in the place representing a reactant s for a, and to check for this we use a transition with an inhibitor arc. However, we also need to ensure that all transitions which feed tokens to s have already had a chance to do so, and we check this using extra activator arcs together with a transition guard which evaluates to true if all such feeding transitions have their local cycle sufficiently high, e.g., transition a_x in Figure 2(b). The overall result for the reaction system $\mathcal{A}_0$ is a high-level net $N_{IV}(\mathcal{A}_0)$ shown in Figure 2(b).

The resulting high-level net is executed according to the standard sequential (interleaving) firing rule and its behaviour closely simulates that of the net obtained by Method III, and so also the behaviour of the original reaction system. We skip the full description of the relationship between these two nets. Intuitively, a marking M of the second translation corresponds directly to a marking of the first one if all the places of the form clk_a contain the same single token k, and all the tokens l in other places satisfy $l \leq k$. (Note that from each reachable marking of the second translation one can execute a sequence of transitions leading to a marking with this property.)

5 Set-nets

In our attempts to obtain a direct and elegant translation from reaction systems into Petri nets, a major and as far as we can tell insurmountable problem was the fact that several transitions may insert tokens into a place representing the presence of a single entity. In this section, we introduce SET-nets, a model that resulted from closer investigations into the possibilities of an OR-treatment of arriving tokens representing the production of entities by reactions. Note that OR-treatment of causality has been considered in [20], but the underlying principle there was completely different from what we are going to propose.

The main idea is that in a SET-net there is no concept of counting. Places are marked or not marked and arcs have no weights. Set-nets resemble elementary net systems (EN-systems) [19] which is a fundamental model to study basic features of concurrent systems, including conflict, causality and independence. However, their execution semantics is different. In SET-nets, a marked place indicates the presence of a resource without any quantification. Hence any number of transitions that take input from this place can be fired at the same time. Moreover, firing a transition empties all its input places. Thus there are no conflicts over tokens in SET-nets, unlike in EN-systems or PT-nets. Similarly, places do not count the tokens, and the firing of a transition simply marks each of its output places (whether or not they were already marked). We will build up the new model in two stages, introducing first SET-nets with only flow arcs.

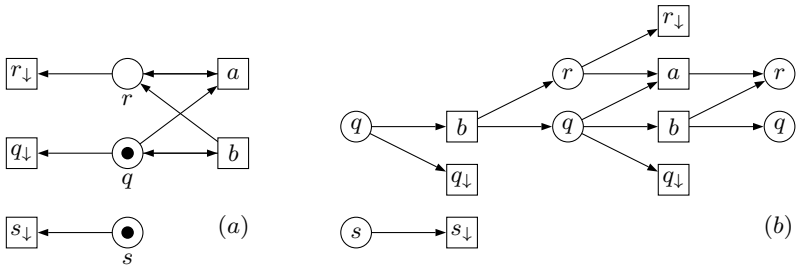


Fig. 3. A SET-net representing reaction system $\mathcal{A}_1$ (a); and an occurrence net constructed for its step sequence $\{b, q_\downarrow, s_\downarrow\}\{a, b, r_\downarrow, q_\downarrow\}$ (b) .

Definition 6 (basic SET-net). *A tuple* $SN = (Pl, Tr, Flw, M_0)$ *is a* (basic) SET-net *if the first three components are as in Definition 4, and* $M_0 \subseteq Pl$ *is the* initial *marking (in general, any set of places is a* marking*).*

The graphical representation of SET-nets is the same as in the case of Petri nets. We now formalise the firing rule for SET-nets.

Definition 7 (marking change). *A set of transitions* U *(also called a* step*) is* enabled *at a marking* M *if* ${}^\bullet U \subseteq M$*. In such a case,* U *can be* fired *with its effect on* M *being given by the resulting marking* $M' = (M \setminus {}^\bullet U) \cup U^\bullet$*. We denote this by* $M[U\rangle M'$*. Moreover, if* U *is the set of* <u>*all*</u> *transitions enabled at* M *(i.e., all transitions* t *satisfying* ${}^\bullet t \subseteq M$*), then we may write* $M[U\rangle_{max} M'$*.*

Hence a step U enabled at a marking M may contain two distinct transitions t and u for which ${}^\bullet t \cap {}^\bullet u \neq \varnothing$ or $t^\bullet \cap u^\bullet \neq \varnothing$ and yet the common places will never contain more than one token. Since *tokens are manipulated using set-based arithmetic* we have chosen the name 'SET-nets' for the new class of Petri nets.

We have introduced first basic SET-nets (without inhibitor and activator arcs), as it seems that one can attempt to develop for them a counterpart of 'structure theory' of PT-nets. To illustrate our point, let us consider a basic SET-net $SN = (Pl, Tr, Flw, M_0)$ with at least one transition. A non-empty set of places $Sphn \subseteq Pl$ is called a *siphon* if ${}^\bullet Sphn \subseteq Sphn^\bullet$. Similarly, a non-empty set of places $Trap \subseteq Pl$ is called a *trap* if $Trap^\bullet \subseteq {}^\bullet Trap$. It can be easily seen that an empty siphon cannot acquire a token by firing any transition, and a marked trap cannot become empty by firing any transition. Both type of sets of places can be used to provide a sufficient condition for deadlock-freeness in PT-nets which was a major motivation behind the development of their structure theory. As it turns out, the same can be done in case of SET-nets.

Theorem 3. *If in the initial marking, every siphon contains a marked trap, then the* SET*-net is deadlock free.*

We next introduce SET-nets with inhibitor and activator arcs.

Definition 8 (SET-net). *A tuple* $SNIA = (Pl, Tr, Flw, Inh, Act, M_0)$ *is a* SET-net *if the first five components are as in Definition 4, and the last one as in Definition 6.*

The definitions and notations concerning the marking change in *SNIA* are the same as for *SN* in Definition 7 with one exception, namely a set of transitions U is enabled at a marking M if ${}^\bullet U \cup {}^\blacklozenge U \subseteq M$ and ${}^\circ U \cap M = \varnothing$. It is interesting to observe that an enabled step U is always *consistent* in the sense that $({}^\bullet U \cup {}^\blacklozenge U) \cap {}^\circ U = \varnothing$. Such a property has a natural and direct (as we will see) connection with the notion of consistency introduced for reaction systems.

As before, given a transition t representing a reaction, the sets ${}^\bullet t$, ${}^\circ t$ and ${}^\blacklozenge t$ correspond to the reactants, inhibitors and products of this reaction. However, we do not require that these sets be non-empty in a SET-net (at least at this point) as such an assumption is not necessary.

6 Reaction systems and SET-nets

Reaction systems and SET-nets fit together well in the sense that both do not count tokens and both change states on the basis of the presence/absence of resources, represented by sets. Moreover, under the SET-net semantics, ordinary arcs (transitions) can be used to empty places. In this semantics, reset arcs with their effect depending on the current number of tokens in a place are meaningless. Finally, following the assumption that all reactions that can take place do take place, the maximal set-semantics can be employed.

Figure 3(a) depicts a SET-net corresponding to a context-independent initialised reaction system $\mathcal{A}_1 = (\{r, q, s\}, \{a, b\}, \{q, s\})$, where $a = (\{r, q\}, \varnothing, \{r\})$ and $b = (\{q\}, \varnothing, \{r, q\})$. (For reasons of clarity, we allow in this section reactions without any inhibitors.) As before, places represent entities. Transitions $r_\downarrow$, $q_\downarrow$ and $s_\downarrow$ ensure that once the SET-net is active only tokens produced in the last maximal step are present in the current marking. For example, we have:

$$\{q, s\}\,[\{b, q_\downarrow, s_\downarrow\}\rangle_{max}\,\{r, q\}\,[\{a, b, r_\downarrow, q_\downarrow\}\rangle_{max}\,\{r, q\}\ ,$$

and so $\sigma = \{b, q_\downarrow, s_\downarrow\}\{a, b, r_\downarrow, q_\downarrow\}$ is a max-step sequence. Relating the behaviour of the SET-net model and the original reaction system is easy and we obtain a counterpart of Theorem 1 with $\nu(M) = M$ and $\nu(U) = U \setminus \{s_\downarrow \mid s \in S\}$.

For a SET-net without inhibitor and activator arcs as in Figure 3(a), one can investigate the causality semantics of reaction systems based on the unfoldings of the corresponding SET-nets. Figure 3(b) shows how such an occurrence net could be derived for the SET-net in Figure 3(a) and its step sequence $\{b, q_\downarrow, s_\downarrow\}\{a, b, r_\downarrow, q_\downarrow\}$ which corresponds of the state sequence $\{b\}\{a, b\}$ of the original reaction system. It is worth observing that the process has branching places which is not possible, in the case of processes of EN-systems or PT-nets. This, however, is fully consistent with the execution semantics of SET-nets.

Modelling inhibition aspects of reactions is rather straightforward using inhibitor arcs, as illustrated by the SET-net in Figure 4(a), representing the context-independent initialised reaction system $\mathcal{A}_2 = (\{r, q, s\}, \{a, b\}, \{q\})$, where:

$$a = (\{r, q\}, \varnothing, \{r\}) \text{ and } b = (\{q\}, \{s\}, \{r, q\}) \text{ and } c = (\{q\}, \varnothing, \{s\}) .$$

Using inhibitor arcs gives a compact translation of reaction systems which is in a sense minimal w.r.t. the number of places, arcs and transitions. Moreover, relating the behaviour of the resulting SET-nets and the original reaction systems can be done as before. Formally, the places, transitions and initial marking of the translation are given by: $Pl = S$, $Tr = A \cup \{s_\downarrow \mid s \in S\}$ and $M_0 = C_0$. There are no activator arcs, and the flow and inhibitor arcs are as follows:

$$\begin{aligned} Flw &= \{(s, s_\downarrow) \mid s \in S\} \cup \{(s, a) \mid a \in A \wedge s \in R_a\} \cup \{(a, s) \mid a \in A \wedge s \in P_a\} \\ Inh &= \{(s, a) \mid a \in A \wedge s \in I_a\} . \end{aligned}$$

The development of a causal process semantics of SET-nets with inhibitor arcs is more difficult. It is therefore interesting to consider models of reaction systems using SET-nets without any inhibitor arcs, as outlined next.

Figure 4(b) shows a SET-net without inhibitor arcs modelling $\mathcal{A}_2$. The way in which it does it is now more involved. More precisely, each execution step of the reaction system is simulated in two phases by the SET-net operating according to the maximal parallelism execution semantics. To keep these two phases clearly separated, they are controlled by an additional cyclic subnet with two places. The key aspect of the construction is the use of a 'complement' s^{cpl} of the 'regular' place s which at the time of checking whether s is empty by reaction b contains a token iff s is empty.

Figure 4(c) provides a generic picture of how, in the proposed construction, a SET-net (without inhibitor arcs) handles an entity r in its role as a reactant, inhibitor, and product. Note that r is represented by two places, r and r^{cpl}, and if r^{cpl} is marked then the entity r in absent in the current state. Moreover, each reaction d is represented by two transitions, d and d'. The first corresponds to the enabling stage of d, and the second to the generation of its products.

The first phase of the simulation always starts in a *consistent* marking M in which there is a token in place phI; for every $s \in S$, $s \in M \Leftrightarrow s^{cpl} \notin M$, and otherwise all places are empty. In this phase transitions corresponding to reactions become active on the basis of the presence and absence of their reactants and inhibitors. Simultaneously, transitions of the form $r_\downarrow$ and $r_\uparrow$ take care that all the entities present in the current state cease to exist (their corresponding places are emptied and the complement places filled). In the second phase, each enabled transition d' finishes the execution of the corresponding reaction, and marks the places corresponding to the entities produced by reaction d and empties their complements.

Relating the behaviour of the SET-net model and the original reaction system is more complicated, using the following two mappings:

$$\nu(M) = M \backslash (\{phI\} \cup \{s^{cpl} \mid s \in S\}) \quad \varphi(U) = U \backslash (\{I\} \cup \{s_\downarrow \mid s \in S\} \cup \{s_\uparrow \mid s \in S\}) .$$

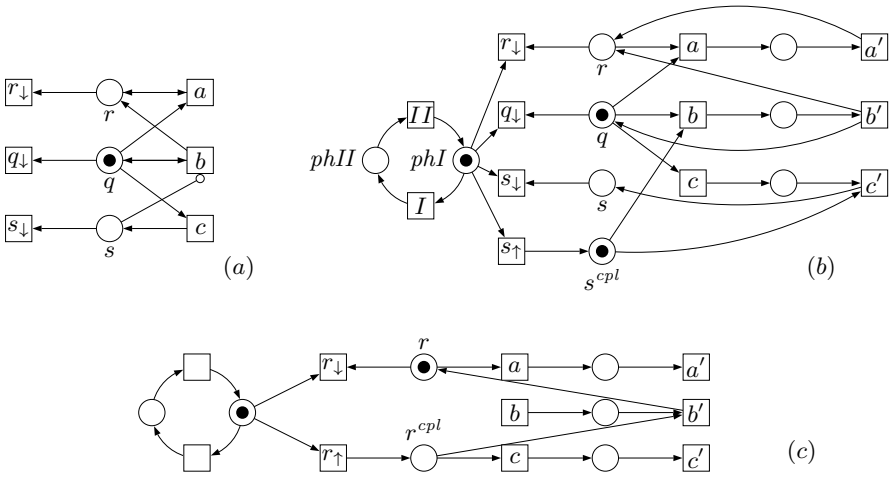


Fig. 4. Two SET-nets representing $\mathcal{A}_2$ (a, b). Generic translation without inhibitor arcs: here r is a reactant for reaction a, product for b, and inhibitor for c (c). Note that not all places and arcs are shown; in particular, each reaction has at least one reactant and hence transitions like c can only fire in the first phase.

One can then show that M_0 is consistent and satisfying $\nu(M_0) = C_0$, and if M is a consistent marking and $M[U\rangle_{max}M''[U'\rangle_{max}M'$ then M' is also consistent.

Theorem 4. *If M is a consistent marking then:*

1. *$M[U\rangle M''[U'\rangle M'$ implies $\nu(M) \stackrel{\varphi(U)}{\Longrightarrow} \nu(M')$.*
2. *$\nu(M) \stackrel{\mathcal{R}}{\Longrightarrow} C$ implies $M[U\rangle M''[U'\rangle M'$ for some U, U', M' and M'' satisfying: $\varphi(U) = \mathcal{R}$ and $\nu(M') = C$*

7 Related work and concluding remarks

When introducing a new class of Petri nets, especially a fundamental one, it is necessary to put it in the context of existing formalisations. To make comparison fair, we will now drop the assumption about maximal parallelism in the execution of SET-nets (which is implied by the execution mode of reaction systems), and consider semantics which allows any set of enabled transitions to be fired.

Set-nets are so simple when it comes to their definition, that it is reasonable to expect that there were in the past net classes with similar features. Indeed, the fundamental class of EN-systems [19] extended with inhibitor as well as activator arcs [12, 17, 18] basically have the same static structure as SET-nets. However, their treatment of conflicts between transitions accessing the same token, as well blocking a transition which could add a token to a marked place, are totally

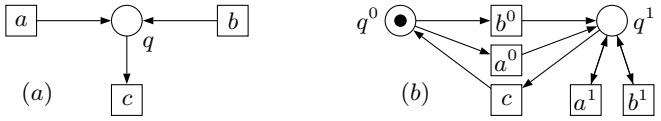


Fig. 5. Boolean net (SET-net with sequential semantics) (a), and 1-safe PT-net simulating its (sequential) behaviour (b).

different. The latter issue has been noted in the past, and the constraint relaxed. For example, there are variations of Petri nets, such as Boolean Petri nets, where adding a token to an already marked place does not add another token [4, 5, 11]. Also, behaviour of this kind was mentioned in [1] in the context of net synthesis. Having said that, the semantics considered in prior works known to us was based on single transition firings, rather than (maximal) steps as is the case for SET-nets. Therefore, the previous models were not concerned with multiple inputs of tokens to a single place something which is essential if one wants to faithfully model reaction systems. Furthermore, by aiming at a set-semantics, we had to introduce the non-conflict feature on the flow arcs consuming the tokens. Therefore, as far as we are aware, the model of SET-nets is an original contribution to the field of Petri nets.

As we already mentioned, SET-nets with interleaving semantics are nothing but Boolean nets used, for instance, in [5]. In such a case, the lack of conflict when firing two transitions sharing an input place is an irrelevant issue, and the only non-standard aspects is that firing a transition with a marked output place does not increase the token count in that place. Such a feature, moreover, can easily be modelled using ordinary 1-safe PT-nets, according to the following idea. First, one splits each place q into places q^0 and q^1, respectively representing the lack and presence of a token in q. Then, each transition t adding tokens to place q is split into t^0 and t^1 to account for two different states the place q can be in represented by q^0 and q^1. Figure 5 illustrates this construction. It can be easily seen that both nets generate the same *sequential* reachability graphs assuming that a^0 and a^1 are instances of a, and b^0 and b^1 are instances of a. However, once we start treating the net in Figure 5(a) as SET-net, the situation changes radically. The reason is that we then have three firings of the following form: $\varnothing[\{a\}\rangle\{q\}$, $\varnothing[\{b\}\rangle\{q\}$ and $\varnothing[\{a,b\}\rangle\{q\}$. Now, the standard classes of Petri nets enjoy the so-called *subset* property which means that if a step U is enabled at marking M, then also any of its subsets is enabled as well. Suppose, then, that there is a Petri net N satisfying this property and such that its step reachability graph is the same as that of the SET-net in Figure 5(a), perhaps after renaming λ being applied to the transitions of the former. Then we have to have two transitions, t and u, in N such that $\lambda(t) = a$, $\lambda(u) = b$ and $M_0[\{t,u\}\rangle M$. Then, by the subset closure property, we also have $M_0[\{t\}\rangle M'$ and $M_0[\{u\}\rangle M''$. Hence, by the reachability graph isomorphism, we must have

$M = M' = M''$ as well as $M_0 \neq M$. Hence we have: $M_0[\{t,u\}\rangle M$ and $M_0[\{t\}\rangle M$ and $M_0[\{u\}\rangle M$ and $M_0 \neq M$. In the standard Petri nets, including various extensions of PT-nets, $M_0[\{t,u\}\rangle M$ and $M_0[\{t\}\rangle M$ would imply that u does not change the current marking. Similarly, $M_0[\{t,u\}\rangle M$ and $M_0[\{u\}\rangle M$ would imply that t does not change the current marking. Yet the simultaneous firing of t and u does change the marking as $M_0 \neq M$. This would produce a contradiction. What we just presented is intuition rather than proof, however, we expect that detailed arguments can be developed for any of the standard net classes. An important consequence, however, is that SET-nets are semantically different from the existing net classes and therefore deserve to be recognised as an original contribution.

8 Conclusions

The main initial motivation of our investigation was to see how Petri net based concepts could be deployed to analyse reaction systems. In particular, we wanted to discover methods for checking properties of reaction systems by relating them to the properties of the corresponding Petri nets and causal processes.

We proposed modelling methods resulting both in low-level and high-level nets. In all four cases, we established a close correspondence between the markings of Petri nets and states of the original reaction systems. The same was true of the evolutions of two corresponding models. In fact, we established that they have essentially isomorphic state spaces. All these net models, however, exhibited deficiencies w.r.t. simplicity and/or elegance and/or tractability of the translation. For example, both high-level net models are intrinsically unbounded, and the second of the low-level translations uses reset arcs. We therefore proposed a new class of Petri nets, called SET-nets, which we feel provide a strong match with the reaction systems and their semantics.

In this way we think we derived new interesting notions and contributions to Petri net theory based on our experiences with reaction systems in a similar way as the concepts of localities and locally maximal concurrency were derived from our previous investigation of a Petri net semantics of membrane systems [15].

Acknowledgement We would like to thank the anonymous reviewers for their suggestions and comments. This research was supported by the Pascal Chair award from Leiden University and the EPSRC VERDAD project.

References

1. E.Badouel and P.Darondeau: Theory of regions. Lecture Notes in Computer Science 1491 (1998) 529–586
2. R.Brijder, A.Ehrenfeucht, M.G.Main and G.Rozenberg: Reaction systems with duration. Lecture Notes in Computer Science 6610 (2011) 191–202
3. R.Brijder, A.Ehrenfeucht, M.G.Main and G.Rozenberg: A Tour of Reaction Systems. Int. Journal of Foundations of Computer Science (2011)

4. L.Czaja: A Calculus of Nets. Cybernetics and Systems Analysis 29 (1993) 185-193
5. P.De Bra, G.J.Houben and Y.Kornatzky: A Formal Approach to Analyzing the Browsing Semantics of Hypertext. Proc. CSN-94 Conference (1994) 78–89
6. C.Dufourd, A.Finkel, and Ph.Schnoebelen: Reset Nets Between Decidability and Undecidability Lecture Notes in Computer Science 1443 (1998) 103-115
7. A.Ehrenfeucht, M.Main and G.Rozenberg: Combinatorics of Life and Death for Reaction Systems. Int. J. of Foundations of Computer Science 22 (2009) 345–356
8. A.Ehrenfeucht and G.Rozenberg: Reaction Systems. Fundamenta Informaticae 76 (2006) 1–18
9. A.Ehrenfeucht and G.Rozenberg: Events and Modules in Reaction Systems. Theoretical Computer Science 376 (2007) 3–16
10. A.Ehrenfeucht and G.Rozenberg: Introducing Time in Reaction Systems. Theoretical Computer Science 410 (2009) 310–322
11. M.Heiner, D.Gilbert and R.Donaldson: Petri Nets for Systems and Synthetic Biology. Lecture Notes in Computer Science 5016 (2008) 215–264
12. R.Janicki and M.Koutny: Semantics of Inhibitor Nets. Information and Computation 123 (1995) 1–16
13. K.Jensen: Coloured Petri Nets and the Invariant-Method. Theoretical Compututer Science 14 (1981) 317-336
14. J.Kleijn and M.Koutny: Processes of Petri Nets with Range Testing. Fundamenta Informaticae 80 (2007) 199–219
15. J.Kleijn, M.Koutny and G.Rozenberg: Process Semantics for Membrane Systems. Journal of Automata, Languages and Combinatorics 11 (2006) 321–340
16. J.Kleijn, M.Koutny and G.Rozenberg: Modelling Reaction Systems with Petri Nets. Technical Report CS-1244. Newcastle University (2011
17. M.Koutny and M.Pietkiewicz-Koutny: Synthesis of Elementary Net Systems with Context Arcs and Localities. Fundamenta Informaticae 88 (2008) 307–328
18. U.Montanari and F.Rossi: Contextual Nets. Acta Informatica 32 (1995) 545–596
19. G.Rozenberg and J.Engelfriet: Elementary Net Systems. Lecture Notes in Computer Science 1491 (1998) 12–121
20. A.Yakovlev, M.Kishinevsky, A.Kondratyev and L.Lavagno: et al: OR Causality: Modelling and Hardware Implementation. Lecture Notes in Computer Science 815 (1994) 568–587

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.53-70

Parameter Estimation of Biological Pathways Using Data Assimilation and Model Checking

Chen Li[1,†], Keisuke Kuroyanagi[2,†], Masao Nagasaki[1,⋆], and Satoru Miyano[1]

[1]Human Genome Center, Institute of Medical Science,
[2]Graduate School of Information Science and Technology, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan
{chenli,ksk9687,masao,miyano}@hgc.jp
http://www.springer.com/lncs

Abstract. This paper presents a novel method to estimate kinetic parameter of biological pathways by using observed time-series data and other knowledge that cannot be formulated in the form of time-series data. Our method utilizes data assimilation (DA) framework and model checking (MC) technique, with a quantitative modeling and simulation architecture named hybrid functional Petri net with extension (HFPNe). Proposed method is applied to an HFPNe model underlying circadian rhythm in *mouse*. We first translate 23 rules of biological knowledge with temporal logic for the model checking, which are not described in the time-series data. Next, we employ particle filter often applied to DA for our estimation procedure. Each particle checks whether its simulation result satisfies the rules or not, and the result of the checking is used for its resampling step. Our simulation results show that proposed method is faster and more accurate than previous method.

Keywords: Hybrid functional Petri net with extension, parameter estimation, data assimilation, particle filter, model checking, temporal logic

1 Introduction

Modeling and simulating large-scale biological pathways have played an important role in systems biology. Owing to their importance, many formal description methods of biological pathway models have been made so far [1–3]. Petri net and its related concepts are one of the succeeded ways of describing biological models [4–6], which have been used for modeling a wide variety of biological pathways and succeeded in reproducing consistent time-series profiles of biological elements such as the concentrations of mRNAs and proteins by means of computer simulations.

Simulation studies on biological pathways promises a deep understanding of complex cellular mechanisms by investigating the dynamic feature. Simulation-based models are commonly governed by a series of parameters, *e.g.* initial values,

⋆ Corresponding author email: masao@hgc.jp. †These authors made equal contributions.

reaction speeds and threshold values of activities. Before the model can be simulated, all parameters must be assigned in advance. However, most parameters are often unknown or not obvious. In general, such parameters are carefully tuned by experts to fit the simulated elements with observed *in vivo/vitro* experiment results. Due to the nonlinearity of the model, parameter estimation is difficult and requires a lot of trial and errors. Small differences of the parameters make large gap between reality and simulation results. Therefore, conventional hand tuning method severely limits the size and complexity of simulation models built as more output data (*e.g.*, microarray gene expression data) are being measured.

The aim of this paper is to develop a novel method to automatically and efficiently estimate kinetic parameters of a given model or a model starting from scratch by combining data assimilation (DA) and model checking (MC) approaches, coupled with observed experiment data. Observed experiment data includes well-defined time-series data and other knowledge that cannot be formulated in the form of time-series data.

DA was originally established in the field of geophysical simulation science. Nagasaki *et al.* [7] have proposed a so-called *genomic data assimilation* approach. Their DA framework enables users to handle both the model construction and parameter tuning in the context of statistical inferences, and establishes a link between the HFPNe simulation model and observed data, *e.g.*, microarray gene expression data [7] or time series proteomic data [8]. Current DA approach has some issues because it depends on providing successive time points (at least 10-20 time points) of time-series data by biological experiments. That is, for a small time-series data set including for example two or three time points, DA will cost massive computational resource, in some cases, it will be completely impossible to estimate parameters. The response to this difficulty of dealing with sparse and/or not well-defined time-series data is the use of model checking.

MC is a method for automatic verification of system requirements [9], which firstly used in hardware field because of its determined behavior and limited value space. Today, this technique has been applied to more complex biological models. There are several studies that use model checking for parameter estimation, *e.g.* Donaldson and Gilbert developed a computational system named MC2(GA) [10] that couples model checking with genetic algorithm for parameter estimation. Li *et al.* [11] proposed online model checking approach based parameter estimation framework applied to the HFPNe class. In order to check the model, one need to write biological rules of the knowledge with temporal logic. For example, "A biological phenomena *Foo* keeps decreasing until *Bar* rises." can be written as "`d([Foo])`$\leq$`0 U d([Bar])>0`" in a kind of temporal logic. Various knowledge can be described in this way. Nevertheless, in order to improve the estimation efficiency and accuracy, it is expected to find a general methodology to determine parameters by combining DA and MC dealing with both time-series experimental data and biological queries.

The paper is organized as follows. In **Methods**, we briefly explain how to (i) construct biological models with HFPNe, (ii) estimate unknown parameters in a nonlinear state space model, (iii) translate biological rules with a temporal logic

for querying system properties, and (iv) combine MC with DA for parameter estimation. In **Results**, we compare our novel method with previous method by applying them to *mouse* circadian rhythm model represented by HFPNe. We show that our method is potentially faster and more accurate than previous one that excludes MC technique.

2 Methods

2.1 Hybrid Functional Petri Net with Extension (HFPNe)

HFPNe is developed as a biosimulation tool for pathway modeling and simulation extended from original Petri net [13]. HFPNe can deal with three types of data: discrete, continuous and generic, whereas the original Petri net deal with only discrete data. HFPNe consists of three types of elements: *entity*, *process* and *connector* (see Figure 1 (a)). Figure 1 (b) shows connection rules in HFPNe. For more definitions and usages of HFPNe, see Nagasaki *et al.* [13].

Figure 2 shows circadian rhythm model of *mouse* represented by HFPNe [14]. This model is composed of of 12 entities, 28 processes and 45 connectors. Due to the space restriction, **Appendix** gives the details of these elements. Initial value of entities $m_i(0)$ (i=1, $\cdots$, 12), reaction speed parameters k_i (i=1, 2 and k_i is a common parameter to control speeds of similar biological processes: k_1 for protein binding; k_2 for translation), and threshold parameters s_i (i=1, $\cdots$, 3) are unknown parameters.

2.2 Data Assimilation for Parameter Estimation

We here explain how to estimate parameters in a simulation model from time-series data with the use of DA.

Data Assimilation with Nonlinear State Space Model DA is an approach to improve the accuracy of the models by combining with data, which can deal with models formulated by nonlinear state space model (SSM) given by two equations [15]:

$$\boldsymbol{m}_t = \boldsymbol{f}(\boldsymbol{m}_{t-1}, \boldsymbol{w}_t, \boldsymbol{\theta}_{sys}) \,, \tag{1}$$

$$\boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{m}_t + \boldsymbol{\epsilon}_t \,. \tag{2}$$

Equation (1) is called "system model" and **Equation** (2) is called "observed model". In the system model, $\boldsymbol{m}_t \equiv (m_{1t}, \cdots, m_{pt})^T$ is a state vector consisting of p state variables m_{it} (i=1, $\cdots$, p) at discrete time point t. $\boldsymbol{w}_t$ denoting system noise is an l-dimensional white noise at t with a density $q(\boldsymbol{w})$. $\boldsymbol{f}$ is a vector-valued function, $\boldsymbol{f} : \mathbb{R}^{p+l} \mapsto \mathbb{R}^p$, and $\boldsymbol{\theta}_{sys}$ is a vector of model parameters. In the observed model, $\boldsymbol{y}_t \in \mathbb{R}^d$ is an observation vector at t, $\boldsymbol{H} \in \mathbb{R}^d \times \mathbb{R}^p$ is an observation matrix; H_{ij} takes value one if observed value of jth entity corresponds to the ith element of $\boldsymbol{y}_t$, otherwise zero. $\boldsymbol{\epsilon}_t$ called observational noise is a d-dimensional white noise at t with a density $r(\boldsymbol{\epsilon})$. The likelihood of the parameter is given by

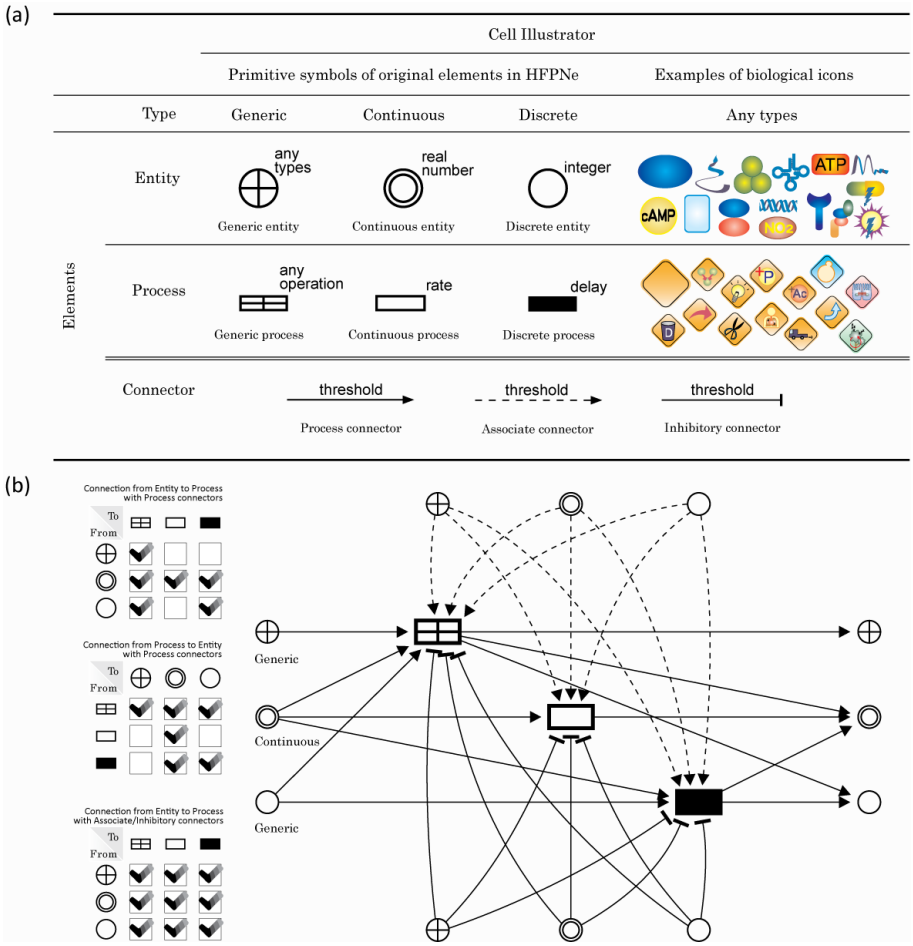


Fig. 1. (a) Basic HFPNe elements and biological icons in Cell Illustrator [12] in which HFPNe was implemented. (b) Connection rules (left side) and corresponding network (right side) in HFPNe. For instance, for the uppermost block labeled with "Connection from Entity to Process with Process connectors", the check-mark denotes the availability connected from corresponding entities to processes, *e.g.*, only the generic process can be selected as the output connected from generic entity with process connector.

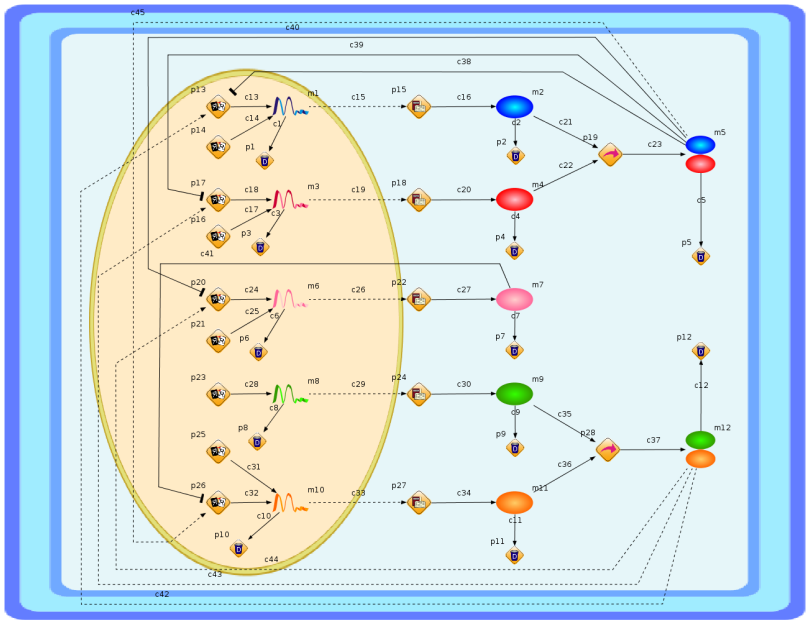


Fig. 2. Circadian rhythm model of *mouse* with HFPNe .

$L(\boldsymbol{\theta}_{sys})=p(\boldsymbol{y}_1,\cdots,\boldsymbol{y}_T|\boldsymbol{\theta}_{sys})=\prod_{t=1}^{T}p(\boldsymbol{y}_t|\boldsymbol{Y}_{t-1},\boldsymbol{\theta}_{sys})=\prod_{t=1}^{T}\int p(\boldsymbol{y}_t|\boldsymbol{m}_t,\boldsymbol{\theta}_{sys})p(\boldsymbol{m}_t|\boldsymbol{Y}_{t-1},\boldsymbol{\theta}_{sys})d\boldsymbol{m}_t$, where $\boldsymbol{Y}_t=\{\boldsymbol{y}_1,\cdots,\boldsymbol{y}_t\}$. The maximum likelihood estimator (MLE) for $\boldsymbol{\theta}_{sys}$ is given by

$$\underset{\boldsymbol{\theta}_{sys}}{\operatorname{argmax}} \log L(\boldsymbol{\theta}_{sys}).$$

We can estimate parameters with MLE, however, this maximum likelihood method has an important issue that the value of $\log L(\boldsymbol{\theta}_{sys})$ computed by Monte Carlo filter includes an approximation error. For accurate estimation, huge computation resources are thus required.

Self-Organizing State Space Model To deal with the difficulty of maximum likelihood method, we use self-organizing state space model (SOSSM) [16–18]. We can estimate parameters based on Bayesian inference with SOSSM by weaving parameters in the state vector as

$$\boldsymbol{z}_t=\begin{bmatrix}\boldsymbol{m}_t\\ \boldsymbol{\theta}_{sys}\end{bmatrix}.$$

The SSM for this vector is given by $\boldsymbol{z}_t=\boldsymbol{F}^*(\boldsymbol{z}_{t-1},\boldsymbol{w}_t)$ and $\boldsymbol{y}_t=\boldsymbol{H}^*\boldsymbol{z}_t+\boldsymbol{\epsilon}_t$, where

$$\boldsymbol{F}^*(\boldsymbol{z}_{t-1},\boldsymbol{w}_t)=\begin{bmatrix}\boldsymbol{f}(\boldsymbol{m}_{t-1},\boldsymbol{w}_t,\boldsymbol{\theta}_{sys})\\ \boldsymbol{\theta}_{sys}\end{bmatrix}$$
$$\boldsymbol{H}^*\boldsymbol{z}_t=\boldsymbol{H}\boldsymbol{m}_t.$$

We can obtain marginal posterior densities without obtaining MLE of $\boldsymbol{\theta}$ as

$$p(\boldsymbol{m}_T \mid \boldsymbol{Y}_T) = \int p(\boldsymbol{z}_T \mid \boldsymbol{Y}_T) d\boldsymbol{\theta}_{sys}$$

$$p(\boldsymbol{\theta}_{sys} \mid \boldsymbol{Y}_T) = \int p(\boldsymbol{z}_T \mid \boldsymbol{Y}_T) d\boldsymbol{m}_T.$$

In this way, we can avoid the issue of maximum likelihood method during the estimation.

Particle Filter As mentioned above, we have to calculate distribution $p(\boldsymbol{z}_T|\boldsymbol{Y}_T)$ for estimating parameters. However, it generally becomes a non-gaussian distribution in SSM. Therefore, it is needed to represent this distribution with some method. We here use a sequential Monte Carlo method called *particle filter* (PF) [19]. Figure 3 shows the overview of particle filter's algorithm. In particle filter, predictive distribution $\boldsymbol{p}(\boldsymbol{z}_t|\boldsymbol{Y}_{t-1})$ and filter distribution $\boldsymbol{p}(\boldsymbol{z}_t|\boldsymbol{Y}_t)$ are approximated by m in which each realization is called *particle* as follows:

$$\boldsymbol{p}_{t|t-1} \equiv \{\boldsymbol{p}_{t|t-1}^{(1)}, \cdots, \boldsymbol{p}_{t|t-1}^{(m)}\} \sim \boldsymbol{p}(\boldsymbol{z}_t \mid \boldsymbol{Y}_{t-1})$$
$$\boldsymbol{p}_{t|t} \equiv \{\boldsymbol{p}_{t|t}^{(1)}, \ldots, \boldsymbol{p}_{t|t}^{(m)}\} \sim \boldsymbol{p}(\boldsymbol{z}_t \mid \boldsymbol{Y}_t),$$

where $\boldsymbol{p}_{t|t-1}^{(j)}$ (j=1, $\cdots$, m) and $\boldsymbol{p}_{t|t}^{(j)}$ (j=1, $\cdots$, m) are (p+$|\boldsymbol{\theta}_{sys}|$)-dimensional numbers. This algorithm is processed with the following steps.

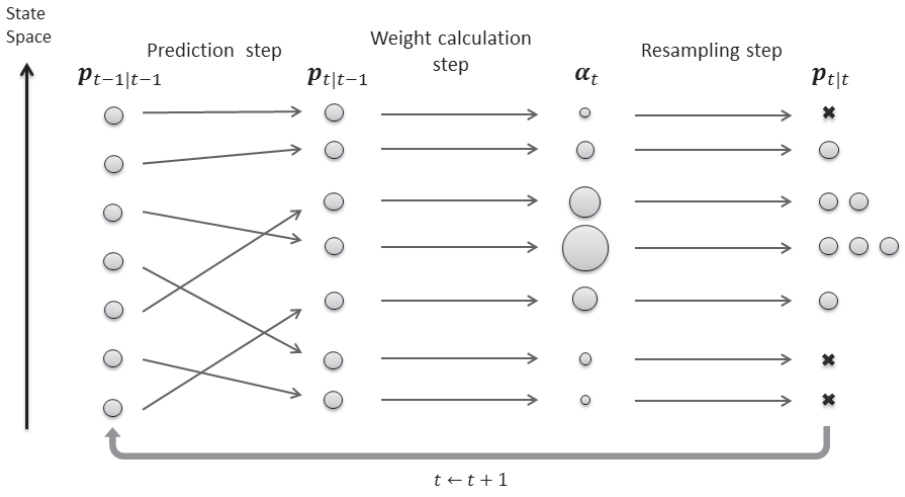


Fig. 3. Overview of particle filter. The left-most column of particles shows state at $t-1$. The second column shows the predicted states. The third column shows the weights of corresponding particles ($\boldsymbol{\alpha}_t \equiv \alpha_t^{(1)}, \cdots, \alpha_t^{(m)}$). The right-most column shows results of resampling step.

Step 1 Generate the $(p+|\boldsymbol{\theta}_{sys}|)$-dimensional random number $\boldsymbol{p}_{0|0}^{(j)}$ for $j=1,\cdots,m$.
Step 2 Repeat the following three steps for the observed time points $t=1,\cdots,T$.

Step 2.1 (Prediction step) Generate m system noises $\boldsymbol{w}_t^{(j)}$ independently and identically from $q(\boldsymbol{w}_t)$, and compute particles by inputting filtered states into simulation model, $\boldsymbol{p}_{t|t-1}^{(j)}=\boldsymbol{F}^*(\boldsymbol{p}_{t-1|t-1}^{(j)},\boldsymbol{w}_t^{(j)})$ for $j=1,\cdots,m$.

Step 2.2 (Weight calculation step) Compute the weights of importance for the particles according to

$$\alpha_t^{(j)}=p(\boldsymbol{y}_t|\boldsymbol{p}_{t|t-1}^{(j)})=r(\boldsymbol{y}_t-\boldsymbol{H}^*\boldsymbol{p}_{t|t-1}^{(j)}) \tag{3}$$

for $j=1,\cdots,m$. These weights are then normalized as $\bar{\alpha}_t^{(j)}=\alpha_t^{(j)}\sum_{j=1}^{m}\alpha_t^{(j)}$.

Step 2.3 (Resampling step) Generate filtered state $\boldsymbol{p}_{t|t}^{(j)}$ for $j=1,\cdots,m$ by resampling $\{\boldsymbol{p}_{t|t-1}^{(1)},\cdots,\boldsymbol{p}_{t|t-1}^{(m)}\}$ with the probabilities $\{\bar{\alpha}_t^{(1)},\cdots,\bar{\alpha}_t^{(m)}\}$.

2.3 Model Checking

We explain how to represent requirements of biological pathway models for model checking. Model checking is a method to verify whether models satisfy the requirements or not. Temporal logic formulae are often used to describe the system requirements. In this study, we selected PLTL (*Probabilistic Linear-time Temporal Logic*) for querying dynamic models of cellular networks [20, 21] which extends original LTL to a stochastic setting with a probability operator and a filter criterion defining the starting state where the property is satisfied.

Table 1. Syntax of PLTL.

ψ	$::= \mathbf{P}_{\trianglelefteq x}(LTL) \mid \mathbf{P}_{=?}(LTL) \mid LTL$
LTL	$::= \phi\{AP\} \mid \phi$
ϕ	$::= \mathbf{X}\phi \mid \mathbf{G}\phi \mid \mathbf{F}\phi \mid \phi\ \mathbf{U}\ \phi \mid \phi\ \mathbf{R}\ \phi \mid \neg\phi \mid \phi\&\&\phi \mid \phi\|\phi \mid \phi \Rightarrow \phi \mid AP$
AP	$::= value\ comp\ value \mid value_{boolean}$
$value$	$::= value\ op\ value \mid [variableName] \mid Function_{numeric} \mid Integer \mid Real$
$value_{boolean}$	$::= true \mid false \mid Function_{boolean}$
$comp$	$::= == \mid != \mid > \mid \geq \mid < \mid \leq$
op	$::= + \mid - \mid * \mid / \mid \hat{}$,
with $\trianglelefteq \in \{<, \leq, >, \geq\}$, $x \in [0,1]$.	

[**PLTL Syntax**] Table 1 shows definition of PLTL syntax, which is used to ask for the probability of user's query via a PLTL formulae ψ. In the LTL expression $\phi\{AP\}$, ϕ will be checked from the state that AP is satisfied rather than from the default initial state, where AP is called *atomic proposition* and takes boolean domain. PLTL allows (i) LTL expression to contain temporal operators, *i.e.*, **X**, **F**, **G**, **U**, **R**. Five temporal operators are used to describe the sequencing of the states along the execution; and (ii) the usage of ψ without probabilistic operators (*i.e.* simply in the form of LTL), which is useful when the model is deterministic.

Table 2. Semantics of temporal operators

Operator	Meaning	Explanation
$\mathbf{X}\phi$	Next time	ϕ must be true at the next time point.
$\mathbf{G}\phi$	Globally	ϕ must always be true.
$\mathbf{F}\phi$	Finally	ϕ must be true at least once.
$\phi_1\mathbf{U}\phi_2$	Until	ϕ_1 must be true until ϕ_2 becomes true; ϕ_2 must become true eventually.
$\phi_1\mathbf{R}\phi_2$	Release	ϕ_2 must be true until and including the time point ϕ_1 becomes true; if ϕ_2 never true, ϕ_1 must always be true.

[**PLTL Semantics**] The semantics of PLTL is defined over the finite sets of finite paths through system's state space, obtained by repeated simulation runs of HFPNe models. The PLTL formula is built upon two components: probabilistic operator and property *LTL*. For each simulation run, the LTL expression is evaluated to a boolean truth value, and the probability of the LTL statement holding true is calculated based on the whole set of simulation results.

For the probability operator components, there are two distinct operators: (i) $\mathbf{P}_{\trianglelefteq x}(LTL)$ is any inequality comparison of the probability of the property LTL holding true, for example $\mathbf{P}_{\geq 0.5}(LTL)$; and (ii) $\mathbf{P}_{=?}(LTL)$ returns the value of the probability of the property holding true. The semantics of the temporal logic operators are described in Table 2. Concentrations of biochemical species in the model are denoted by [*variableName*]. A special variable, [*time*], stands for simulation time.

Due to the ability of PLTL, it is possible to define functions of two different natures: functions that return a real number and functions that return a boolean value. An example of the real number function is $d([variableName])$ which returns the subtracted value of [*variableName*] between time i and $i-1$. Note that, d([*variableName*]) equals zero at time point zero. One example of a boolean function is $similarAbsolute(value\ a, value\ b, value\ \epsilon)$, which returns true if $|a-b|<\epsilon$ or else it returns false. Table 3 shows the rules written in PLTL for circadian rhythm model of *mouse*.

2.4 Combining Model Checking with DA

As stated in **Introduction**, we have applied either DA or MC to pathway model in the previous researches, but used time scales are different. We here employ common time scale called *Petri net time* for combining DA with MC, which is the virtual time unit of the HFPNe model denoted by [pt]. We define: (i) $simInt \in \mathbb{R}$ as simulation interval; (ii) $mcInt \in \mathbb{R}$ which is a multiple of $simInt$ as a model checking interval; and (iii) $MapOttoPt : \mathbb{N} \mapsto \mathbb{R}$ as a mapping from observed time point to Petri net time for combining. From the MC's viewpoint, $\mathbf{X}\phi$ means that ϕ must be true at the state after $mcInt$ Petri net time. Meanwhile, a simulation from time $MapOttoPt(t-1)$ to $MapOttoPt(t)$ is a run in $\boldsymbol{f}$ from DA's viewpoint.

Hürseler and Künsch mentioned that it is difficult to generate a good initial distribution of parameters with SOSSM [22]. This is because parameters in resampled particles are the subset of parameters in the initial particles and a model with randomly generated parameters rarely satisfy all rules. To generate good initial distribution of parameters, two considerations are designed:

Table 3. Biological rules for circadian rhythm model of *mouse*. Rule 1 to Rule 9 describes the range of concentrations; Rule 10 to Rule 17 describes that of peak concentrations, Rule 18 to Rule 21 specifies concentration relationships when they reach peaks, and Rule 22 to Rule 23 specifies normal concentration relationships.

No.	Rule LTL translation
Rule 1	Concentration of per_mRNA is between 0.2 and 3.8. `G([per_mRNA] < 3.8 && [per_mRNA] > 0.2)`
Rule 2	Concentration of Rev-Erv_mRNA is between 0.2 and 3.4. `G([Rev-Erv_mRNA] < 3.4 && [Rev-Erv_mRNA] > 0.2)`
Rule 3	Concentration of Bmal_mRNA is between 0.2 and 4.3. `G([Bmal_mRNA] < 4.3 && [Bmal_mRNA] > 0.2)`
Rule 4	Concentration of Bmal_mRNA is between 2.4 and 2.6 after time becomes 20. `G(similarAbsolute([Clock_mRNA],2.5,0,1)){[time] > 20}`
Rule 5	Concentration of Cry_mRNA is between 3.8 and 0.2. `G([Cry_mRNA] < 3.8 && [Cry_mRNA] > 0.2)`
Rule 6	Concentration of PER is between 2.6 and 0.2. `G([PER] < 2.6 && [PER] > 0.2)`
Rule 7	Concentration of CRY is between 2.6 and 0.2. `G([CRY] < 2.2 && [CRY] > 0.2)`
Rule 8	Concentration of PER/CRY is between 2.7 and 0.2. `G([PER/CRY] < 2.7 && [PER/CRY] > 0.2)`
Rule 9	Concentration of REV_ERB is between 1.5 and 0.2. `G([REV_ERB] < 1.5 && [REV_ERB] > 0.2)`
Rule 10	Local maximum concentration of per_mRNA is greater than 2.0. `G(d([per_mRNA]) ≥ 0 && X(d([per_mRNA])<0) ⇒ [per_mRNA] > 2.0)`
Rule 11	Local minimum concentration of per_mRNA is less than 1.0. `G(d([per_mRNA])≤0 && X(d([per_mRNA])>0)⇒ [per_mRNA] < 1.0)`
Rule 12	Local maximum concentration of Rev-Erv_mRNA is greater than 1.5. `G(d([Rev-Erv_mRNA])≥0 && X(d([Rev-Erv_mRNA])<0) ⇒ [Rev-Erv_mRNA] > 1.5)`
Rule 13	Local minimum concentration of Rev-Erv_mRNA is less than 1.0. `G(d([Rev-Erv_mRNA])≤0 && X(d([Rev-Erv_mRNA])>0) ⇒ [Rev-Erv_mRNA] < 1.0)`
Rule 14	Local maximum concentration of Bmal_mRNA is greater than 1.5. `G(d([Bmal_mRNA]) ≥0 && X(d([Bmal_mRNA])<0) ⇒ [Bmal_mRNA] > 1.5)`
Rule 15	Local minimum concentration of Bmal_mRNA is less than 1.0. `G(d([Bmal_mRNA])≤0 && X(d([Bmal_mRNA])>0) ⇒ [Bmal_mRNA] < 1.0)`
Rule 16	Local maximum concentration of Cry_mRNA is greater than 2.0. `G(d([Cry_mRNA])≥0 && X(d([Cry_mRNA])<0) ⇒ [Cry_mRNA] > 2.0)`
Rule 17	Local minimum concentration of Cry_mRNA is less than 1.0. `G(d([Cry_mRNA])≤0 && X(d([Cry_mRNA])>0) ⇒ [Cry_mRNA] < 1.0)`
Rule 18	When concentration of Bmal_mRNA takes local minimum, concentration of Bmal_mRNA is less than concentration of per_mRNA. `G(d([Bmal_mRNA])≤0 && X(d([Bmal_mRNA])>0) ⇒ [Bmal_mRNA] < [per_mRNA])`
Rule 19	When concentration of Bmal_mRNA takes local maximum, concentration of Bmal_mRNA is grater than concentration of per_mRNA `G(d([Bmal_mRNA])≥0 && X(d([Bmal_mRNA])<0) ⇒ [Bmal_mRNA] > [per_mRNA])`
Rule 20	When concentration of per_mRNA takes local minimum, concentration of per_mRNA is less than concentration of per_mRNA. `G(d([per_mRNA])≤0 && X(d([per_mRNA])>0) ⇒ [per_mRNA] < [Bmal_mRNA])`
Rule 21	When concentration of per_mRNA takes local maximum, concentration of per_mRNA is grater than concentration of Bmal_mRNA `G(d([per_mRNA])≥0 && X(d([per_mRNA])<0) ⇒ [per_mRNA] > [Bmal_mRNA])`
Rule 22	Concentration of per_mRNA is greater than concentration of Rev-Erv_mRNA. `G([per_mRNA] > [Rev-Erv_mRNA])`
Rule 23	Concentration of CLOCK is greater than concentration of PER. `G([CLOCK] > [PER])`

Firstly, we repeat particle filters many times by regenerating $\boldsymbol{p}_{0|0}$ from $\boldsymbol{p}_{T|T}$ like a crossover in genetic algorithm since the length of time courses for biological use is generally shorter than that for other fields; Secondly, a threshold value Th is used to discard particle whose unsatisfied number by the checking is greater than Th. Th is changed for each run of particle filter. In this study, system noise is not taken into account in order to accelerate the estimation. Nevertheless, a generic process can be mapped to Java object supporting HFPNe model in a nondeterministic settings.

Algorithm 1 shows pseudocode of our parameter estimation method. In step 9, $PredictandMC$ returns predicated particle and the result whether it is worthless or not. The detail of this function is displayed in **Algorithm 2**. In steps 10–15, if the simulation results unsatisfy more than Th rules, the weight of the particle will become zero, or else the weight is calculated by **Equation** (3) via r. To calculate r, we assume that observed noise is a gaussian white noise and its mean is zero. The variance of observed noise is thus needed to be estimated. We generate multiple candidate values and use the value that has the maximum likelihood as the variance. Steps 17–19 are designed to break the run of particle filter if all the particles unsatisfied more than Th rules; Otherwise Th is incremented or decremented in steps 23–27. $\boldsymbol{p}_{0|0}$ for the next run of particle filter is generated from $\boldsymbol{p}_{T|T}$ in step 28.

Algorithm 1. Pseudecode of our parameter estimation method.

1: **function** $EstimateParameters(\boldsymbol{Y}_T, Rules)$
2: $\boldsymbol{p}_{0|0} \leftarrow GenerateInitialParticles()$
3: $Th \leftarrow initialThreshold$
4: $F \leftarrow false$
5: **loop**
6: **for** $t \leftarrow 1, \ldots, T$ **do**
7: $F \leftarrow false$
8: **for** $j \leftarrow 1, \ldots, m$ **do**
9: $(\boldsymbol{p}^{(j)}_{t|t-1}, SatisfiedRules) \leftarrow PredictandMC(\boldsymbol{p}^{(j)}_{t-1|t-1}, t, Rules, Th)$
10: **if** $SatisfiedRules$ **then**
11: $\alpha^{(j)}_t \leftarrow CalculateWeight(\boldsymbol{p}^{(j)}_{t|t-1}, \boldsymbol{y}_t)$
12: $F \leftarrow true$
13: **else**
14: $\alpha^{(j)}_t \leftarrow 0$
15: **end if**
16: **end for**
17: **if** $!F$ **then**
18: **break**
19: **end if**
20: $\bar{\boldsymbol{\alpha}}_t \leftarrow \boldsymbol{\alpha}_t / \sum_{j=1}^{M} \alpha^{(j)}_t$
21: $\boldsymbol{p}_{t|t} \leftarrow Resampling(\boldsymbol{p}_{t|t-1}, \bar{\boldsymbol{\alpha}}_t)$
22: **end for**
23: **if** F **then**
24: $Th \leftarrow \max(0, Th - 1)$
25: **else**
26: $Th \leftarrow Th + 1$
27: **end if**
28: $\boldsymbol{p}_{0|0} \leftarrow RegenerateParticles(\boldsymbol{p}_{T|T})$
29: **end loop**

In **Algorithm 2**, steps 2–3 convert time scale from observed time to Petri net time. Step 4 separates particle $\boldsymbol{p}$ into $\boldsymbol{m}(start)$ and $\boldsymbol{\theta}$, where $\boldsymbol{m}(t)$ is a vector consisting of the values of p entities at Petri net time t and $\boldsymbol{\theta}$ is a vector of parameters to be estimated. $\boldsymbol{m}(pt{+}simInt)$ is predicted by simulation in step 6. Step 7 returns the number of unsatisfied rules via $ModelChecking()$. In steps 9–11, if simulation result violates more than Th rules, simulation will stop immediately.

Algorithm 2 Prediction step of particle filter and model checking.

1: **function** $PredictandMC(\boldsymbol{p}, t, Rules, Th)$
2: $start \leftarrow MapOttoPt(t-1)$
3: $end \leftarrow MapOttoPt(t) - simInt$
4: $(\boldsymbol{m}(start), \boldsymbol{\theta}) \leftarrow \boldsymbol{p}$
5: **for** $pt \leftarrow start$ **to** end **step** $simInt$ **do**
6: $\quad \boldsymbol{m}(pt + simInt) \leftarrow Simulation(\boldsymbol{m}(pt), \boldsymbol{\theta})$
7: $\quad$ **if** $(pt + simInt)\%mcInt = 0$ **then**
8: $\quad\quad N \leftarrow ModelChecking(\boldsymbol{m}, Rules)$
9: $\quad\quad$ **if** $Th < N$ **then**
10: $\quad\quad\quad$ **return** $([\boldsymbol{m}(pt + simInt)^T, \boldsymbol{\theta}^T]^T, false)$
11: $\quad\quad$ **end if**
12: $\quad$ **end if**
13: **end for**
14: **return** $([\boldsymbol{m}(MapOttoPt(t))^T, \boldsymbol{\theta}^T]^T, true)$

3 Results and Discussions

3.1 Estimation environment and evaluation criteria

We estimate 17 parameters of circadian rhythm model of *mouse* (*i.e.*, 12 initial values of entities, two reaction speeds and three threshold values). We use synthesized data set coupled with simulation and observed noise $\boldsymbol{\epsilon}_t$ ($\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, 0.05^2 I_{12})$) as observed data. It contains 312 data of 26 time points (see Figure 4) for each 12 biological entities – five mRNAs, five proteins and two complex proteins –. One observed time point is mapped to five Petri net times. Table 4 summarizes details of our estimation environment. Parameter search range is set from zero to 15.

Table 4. Default parameters of the estimation.

Parameter name	Value	Meaning
m	50,000	the number of particles
p	12	the number of entities
d	12	the number of entities which have observed data
T	26	the number of observed time points
$simInt$	0.1	simulation interval
$mcInt$	1.0	model checking interval
$initialThreshold$	8	initial value of threshold for model checking

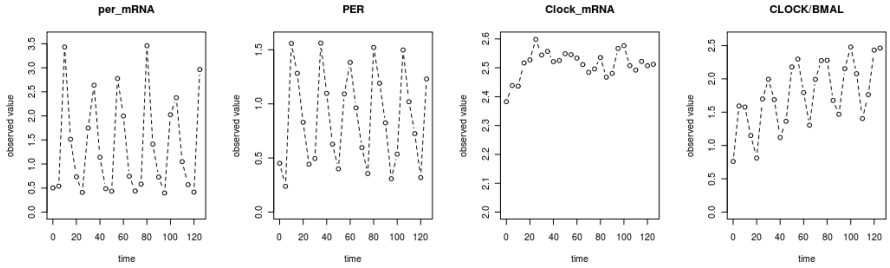


Fig. 4. Examples of observed data.

To evaluate the results of estimation, following *Score*s are defined and calculated before the step *RegenerateParticles* step given in **Algorithm 1**.

$$Score_{mean} = \sum_{t=1}^{T}\sum_{e=1}^{p} \frac{|SimResult(Params_{mean},e,t)-y_{t,p}|^2}{|\boldsymbol{Y}_T|}$$
$$Score_{mode} = \sum_{t=1}^{T}\sum_{e=1}^{p} \frac{|SimResult(Params_{mode},e,t)-y_{t,p}|^2}{|\boldsymbol{Y}_T|}$$
$$Score_{median} = \sum_{t=1}^{T}\sum_{e=1}^{p} \frac{|SimResult(Params_{median},e,t)-y_{t,p}|^2}{|\boldsymbol{Y}_T|}$$
$$Score_{best} = \sum_{t=1}^{T}\sum_{e=1}^{p} \frac{|SimResult(Params_{best},e,t)-y_{t,p}|^2}{|\boldsymbol{Y}_T|}$$
$$Score = \min\{Score_{mean}, Score_{mode}, Score_{median}, Score_{best}, Score_{current}\},$$

where $y_{t,p}$ is an observed value of entity p at observed time point t. $Params_{best}$ denotes parameters of a particle which is made of particles. $Params_{mean}$, $Params_{mode}$ and $Params_{median}$ represent mean, mode and median value of all particles' parameters respectively. $SimResult(Params, e, t)$ returns the value of entity e at time t which is calculated by simulation with $Param$. $Score_{current}$ is ∞ at first time of the calculation, otherwise it is the *Score* of previous calculation.

3.2 Experimental Result

Comparison between PFMC and PF We compare the performance between our new method Particle Filter with Model Checking (PFMC) and previous method Particle Filter (PF). The estimation experiments are carried out on workstation of Intel Xeon E5450 (3.0GHz) with 32G bytes of memory. We performed estimation 100 times for both methods. Figure 5 shows the result with respect to the distribution of *Score* with elapsed time. Mean of *Score* is also analyzed by Welch's t-test (See Table 5). Nearly all *Score* of PFMC and PF are good on and after 600 seconds. Both medians are also good on and after 600 seconds, but there are many bad cases before 1,000 seconds for PF. That is, roughly speaking, if there are enough amounts of observed data, there is no much difference by using either PMFC or PF for the estimation. However, in almost all cases, we can finish the estimation within a short time incorporating model checking.

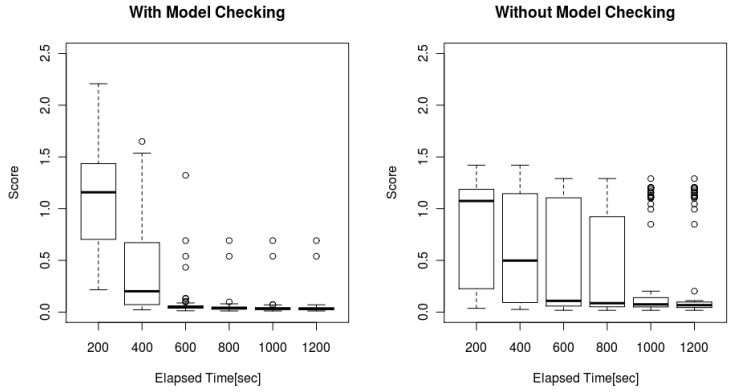


Fig. 5. Distributions of *Score* at corresponding time points. The left diagram shows the result of PFMC, and the right one shows that of PF. X-axis denotes elapsed time (second), while y-axis denotes *Score*.

Table 5. Results of Welch's t-test.

Elapsed time[sec]	P-value	Mean of PFMC *Score*	Mean of PF *Score*
200	1.584×10^{-6}	1.1055600	0.7746972
400	0.02169	0.4656204	0.5243369
600	1.834×10^{-10}	0.07760144	0.44547495
800	6.165×10^{-9}	0.0516071	0.3580496
1000	9.631×10^{-8}	0.04642736	0.31403475
1200	2.974×10^{-7}	0.04446433	0.29820645

Estimation with small amount of observed data To investigate the performance in the case of small amount of observed data, we use only first ten time points of five biological entities' data for the estimation: per_mRNA, Cry_mRNA, Rev_Erv_mRNA, $Clock\_mRNA$ and $Bmal\_mRNA$. More detailedly, we use all default parameters for *Score* calculation and just overwrite p to five and T to ten. The estimation results are exhibited in Figure 6 and Table 6.

The results clearly show that on and after 1,200 seconds, in contrast to the previous experiment, there is difference not only between bad cases, but also between medians of PFMC and PF. This is because it is difficult to estimate parameters which makes certain rhythms with only two cycles of observed data. Therefore, median *Score*s of PF are not good before 3,000 seconds. Moreover, convergence of PFMC is worse than previous experiment.

Effects of the rules We also investigate the effect of rules by checking which rule is unsatisfied which results in the cutting of a particle. We run estimation 100 times with default parameters and all of observed data. All the runs finished after 20 minutes and the results are shown in Figure 7.

Two kinds of effects used in model checking approach can be considered. First, it cuts useless particles,which enables us to estimate with more particles or more runs of particle filters. Second, it cuts bad results which facilitates the

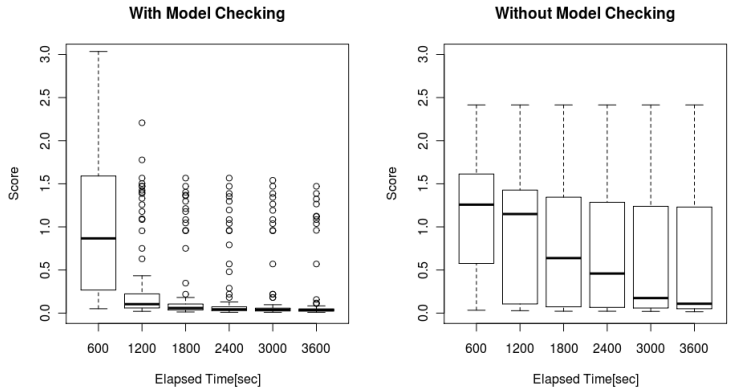


Fig. 6. Distributions of *Score* with small amount of observed data.

Table 6. Results of Welch's t-test.

Elapsed time[sec]	P-value	Mean of PFMC Scores	Mean of PF Scores
600	0.3340	6.401179	1.137218
1200	3.725×10^{-10}	0.3219618	0.8769369
1800	1.393×10^{-10}	0.2144606	0.7418111
2400	1.797×10^{-10}	0.1849873	0.6784018
3000	8.596×10^{-9}	0.1677461	0.6048147
3600	3.071×10^{-8}	0.1469600	0.5597454

estimation only with observed data. From the first effect's viewpoint, the rule that cut more particles is a good rule. This is due to the fact that rule is able to cut many particles from early time because the number of unique particle decreases with the time elapse in our method. Rule 1 to 3, 5 to 9, 22 and 23 are such rules. From the second effect's viewpoints, good rules are different depending on behaviors of target models. For the circadian rhythm model, it is important to reproduce the oscillations within a certain range. Rule 10 to 21 are specified for verifying the behavior of oscillation. Generally, it is not easy to prepare this kind of rules before trial. Nevertheless, unlike observed data, it will be a great help in improving the efficiency and accuracy of conventional parameter estimation process and eventually leading to better understanding of biological pathways.

4 Conclusions

We propose a novel parameter estimation method for biological pathways. By combining model checking with DA framework, our method enables us to use various knowledge in addition to observed time series data. We extract 23 biological rules with temporal logic which cannot be formulated in the form of time-series data. Proposed method and previous method are applied to *mouse* circadian rhythm model of HFPNe by means of performance evaluation. Results

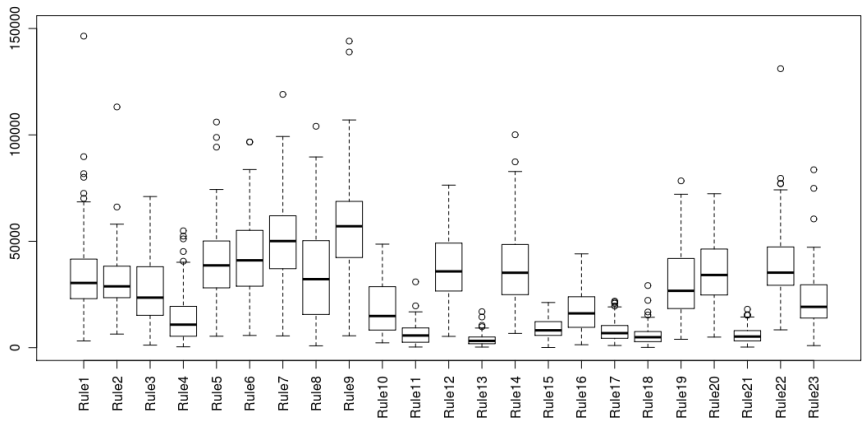


Fig. 7. The number of the particle cutting happened by model checking with respect to each rule. X-axis denotes rule number, and y-axis denotes the number of the cutting.

shows that (i) if estimations execute with enough amounts of observed data, our method can practically give good parameters in a short time; and (ii) if estimations execute with small amount of observed data, new method is much faster than the method without model checking.

References

1. Yan, H., Zhang, B., Li, S., Zhao, Q.: A Formal Model for Analyzing Drug Combination Effects and its Application in TNF-alpha-induced NFkappaB Pathway. BMC Syst Biol., 4(50) (2010)
2. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model Building and Model Checking for Biochemical Processes. Cell Biochem. Biophys., 38(3), 271–286 (2003)
3. Regev, A., Silverman, W., Shapiro, E.: Representation and Simulation of Biochemical Processes Using the Pi-calculus Process Algebra. Pac. Symp. Biocomput., 459–470 (2001)
4. Chaouiya, C.: Petri Net Modelling of Biological Networks. Brief Bioinform., 8(4), 210–219 (2007)
5. Koch, I., Heiner, M.: Petri nets. In Analysis of Biological Networks. Edited by Junker, B.H., Schreiber, F. A Wiley Interscience Publication, 139–180 (2008).
6. Matsuno, H., Li, C., Miyano, S.:Petri Net Based Descriptions for Systematic Understanding of Biological Pathways. IEICE Trans. Fundamentals, E89-A(11), 3166–3174 (2006)
7. Nagasaki, M., Yamaguchi, R., Yoshida, R., Seiya, I., Doi, A., Tamada, Y., Matsuno, H., Miyano, S.: Genomic Data Assimilation for Estimating Hybrid Functional Petri Net from Time-course Gene Expression Data. Genome Inform., 17(1), 46–61, 2006.

8. Tasaki, S., Nagasaki, M., Kozuka-Hata, H., Semba, K., Gotoh, N., Hattori, S., Inoue, J., Yamamoto, T., Miyano, S., Sugano, S., Oyama, M.: Phosphoproteomics-based Modeling Defines the Regulatory Mechanism Underlying Aberrant EGFR Signaling. PLoS One, 5(11), e13926 (2010)
9. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P., McKenzie, P.: Systems and Software Verification: Model-Checking Techniques and Tools, Springer (2001)
10. Donaldson, R., Gilbert, D.: A Model Checking Approach to the Parameter Estimation of Biochemical Pathways. In: Proceedings of the 6th International Conference on Computational Methods in Systems Biology (CMSB '08), pp. 269–287. Springer-Verlag, Berlin, Heidelberg (2008).
11. Li, C., Nagasaki, M., Hock Koh, C., Miyano, S.: Online model checking approach based parameter estimation to a neuronal fate decision simulation model in C. elegans with hybrid functional Petri net with extension, Mol. Biosyst., 7(5), 1576–1592 (2011)
12. Nagasaki, M., Saito, A., Jeong, E., Li, C., Kojima, K., Ikeda, E., Miyano, S.: Cell Illustrator 4.0: A Computational Platform for Systems Biology, In Silico Biol., 10, 0002 (2010)
13. Nagasaki, M., Doi, A., Matsuno, H., Miyano, S.: A Versatile Petri Net Based Architecture for Modeling and Simulation of Complex Biological Processes. Genome Inform., 15(1), 180–197 (2004)
14. Circadian rhythms in Mus musculus, `http://www.csml.org/models/csml-models/circadian-rhythms-in-mouse/`
15. Kitagawa, G.: Non-Gaussian State-space Modeling of Nonstationary Time Series. Journal of the American Statistical Association, 82(400), 1032–1063, 1987.
16. Kitagawa, G.: Self-organizing State Space Model. J. of the American Statistical Association, 93, 1203–1215 (1998)
17. Higuchi, T.: Self-organizing Time Series Model. In Sequential Monte Carlo Methods in Practice, Springer-Verlag New York, 429–444 (2001)
18. Kitagawa, G. and Sato, S.: Monte Carlo Smoothing and Self-Organising State-space Model. In Sequential Monte Carlo Methods in Practice, Springer-Verlag New York, 177–195 (2001)
19. Kitagawa, G.: Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models.Journal of Computational and Graphical Statistics, 5(1), 1–25 (1996)
20. Heiner, M., Lehrack, S., Gilbert, D., Marwan, W.: Extended Stochastic Petri Nets for Model-based Design of Wetlab Experiments. Edited by Priami, C., et al., Trans. on Comput. Syst. Biol. XI, LNBI, 5750, 138–163 (2009)
21. Donaldson, R., Gilbert, D.: A Monte Carlo Model Checker for Probabilistic LTL with Numerical Constraints, Technical report, University of Glasgow, Department of Computing Science (2008)
22. Hürzeler and Hans, M., Künsch, R.: Approximating and maximising the likelihood for a general state-space model. In: Sequential Monte Carlo Methods in Practice, Springer-Verlag New York, 159–175 (2001)

Appendix:

Table 7. Biological entities in the model of Figure 2. Variable $m_i(t)$ $(i = 1, \ldots, 12)$ indicates the concentration of corresponding entity at time t. $m_i(0)$ $(i = 1, \ldots, 12)$ is the initial value of corresponding entity.

Entity Name	Variable	Initial Value	Biological Type
per_mRNA	$m_1(t)$	$m_1(0)$	mRNA
PER	$m_2(t)$	$m_2(0)$	protein
Cry_mRNA	$m_3(t)$	$m_3(0)$	mRNA
CRY	$m_4(t)$	$m_4(0)$	protein
PER/CRY	$m_5(t)$	$m_5(0)$	complex protein
Rev-Erv_mRNA	$m_6(t)$	$m_6(0)$	mRNA
REV-ERV	$m_7(t)$	$m_7(0)$	protein
Clock_mRNA	$m_8(t)$	$m_8(0)$	mRNA
CLOCK	$m_9(t)$	$m_9(0)$	protein
Bmal_mRNA	$m_{10}(t)$	$m_{10}(0)$	mRNA
BMAL	$m_{11}(t)$	$m_{11}(0)$	protein
CLOCK/BMAL	$m_{12}(t)$	$m_{12}(0)$	complex protein

Table 8. Processes and their reaction speeds in the model. (k_1 and k_2 are common parameters to control speeds of similar biological processes: k_1 for protein binding. k_2 for translation.

Process Name	Biological Process Type	Speed of corresponding processes
$p1$	degradation	$v_1(t) = m_1(t) \times 0.2$
$p2$	degradation	$v_2(t) = m_2(t)/7$
$p3$	degradation	$v_3(t) = m_3(t) \times 0.2$
$p4$	degradation	$v_4(t) = m_4(t) \times 0.1$
$p5$	degradation	$v_5(t) = m_5(t)/15$
$p6$	degradation	$v_6(t) = m_6(t) \times 0.2$
$p7$	degradation	$v_7(t) = m_7(t) \times 0.1$
$p8$	degradation	$v_8(t) = m_8(t) \times 0.2$
$p9$	degradation	$v_9(t) = m_9(t)/7$
$p10$	degradation	$v_{10}(t) = m_0(t) \times 0.2$
$p11$	degradation	$v_{11}(t) = m_1(t) \times 0.1$
$p12$	degradation	$v_{12}(t) = m_2(t)/15$
$p13$	transcription	$v_{13}(t) = 1$
$p14$	transcription	$v_{14}(t) = 0.05$
$p15$	translation	$v_{15}(t) = m_1(t)/k_2$
$p16$	transcription	$v_{16}(t) = 0.05$
$p17$	transcription	$v_{17}(t) = 1$
$p18$	translation	$v_{18}(t) = m_3(t)/k_2$
$p19$	binding	$v_{19}(t) = m_2(t) \times m_4(t)/k_1$
$p20$	transcription	$v_{20}(t) = 1$
$p21$	transcription	$v_{21}(t) = 0.05$
$p22$	translation	$v_{22}(t) = m_6(t)/(2 \times k_2)$
$p23$	transcription	$v_{23}(t) = 0.5$
$p24$	translation	$v_{24}(t) = m_8(t)/k_2$
$p25$	transcription	$v_{25}(t) = 0.05$
$p26$	transcription	$v_{26}(t) = 1.1$
$p27$	translation	$v_{27}(t) = m_{10}(t)/k_2$
$p28$	binding	$v_{28}(t) = m_9(t) \times m_1(t)/k_1$

Table 9. Threshold parameters s_i $(i = 1, \ldots, 3)$ and the corresponding connectors in the model.

Threshold parameters	Name of regulation	Corresponding connector
s_1	Bmal_mRNA active regulation	$c45$
s_2	Rev-Erv_mRNA inhibitory regulation	$c40$
s_3	Cry_mRNA inhibitory regulation	$c39$

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.71-86

Studying Steady States in Biochemical Reaction Systems by Time Petri Nets

Louchka Popova-Zeugmann[1] and Elisabeth Pelz[2]

[1] Department of Computer Science, Humboldt University, Berlin, Germany,
popova@informatik.hu-berlin.de
[2] LACL, University Paris Est Créteil, Fac de Sciences, F- 94010 Créteil
pelz@u-pec.fr

Abstract. Biochemical reaction systems are usually modeled by ordinary differential equations (ODEs). For further analysis, they are often transformed into stochastic Petri nets (SPN), whose state space (or reachability graph) then can be studied to deduce properties.
If a biochemical reaction system is in a steady state, from now on called **steady situation**[1], then the rates of the reactions and the concentrations of the species are constant. These concentrations and rates can be established by simulation of the SPN-model. A steady situation[1] signifies also that on the model level only a subset of all possible reachable states is pertinent to this situation. It would be of interest to isolate formally and constructively this subset of states. To our knowledge there is no way to achieve this using the SPN-model or the ODE-model.
In this article we propose an approach to calculate the part of the state space corresponding to a steady situation[1]. To do so, we map the SPN-model onto a Time Petri Net-model (TPN) with the same behaviour as that in the steady situation[1] observed in the SPN simulation.
Using reduction methods for TPNs we can extract the part of the reachability graph of the SPN-model which is relevant for the steady-situation[1]. We show that this is exactly the reduced reachability graph of the constructed TPN-model. Finally, the later one can be analyzed qualitatively and quantitatively.
In addition, this approach helps for validating the correctness of the calculated (and used) rates in the steady situation[1]and of the parameters used in the original ODEs, fixed by experiments in the wet labs, both being -a priori- subject to a certain degree of uncertainty.

1 Introduction

When considering biochemical reaction systems we emphasize the interactions between different species during time and we do not take a momentary snapshot of the system. This means that time is an indispensable component in each

[1] to avoid confusion, between **state** in the biochemical system and **states** or **state space** in the models, the biological **steady state** will be called throughout the whole paper **steady situation**

model of such a system. Furthermore, the repetitive occurrence of reactions in the system during a certain time, expressed by their reaction rates, defines the behaviour of the system. It is obvious, that the rates depend on the concentrations of the species involved in the reactions: the higher the concentration the higher the reaction rate. This is the case until the concentrations of the involved species achieve certain levels. Then the concentrations of the species do no longer change, i.e., the reaction rates stay constant. This situation is the so called steady situation[1] in an biochemical reaction system. Finally, the occurrence (or taking place) of biochemical reactions is a stochastical one.

The taking place of a particular reaction can be modeled by an ordinary differential equation (ODE), the causal relationship between the interactions is often modeled by some graph. Both aspects can be represented in a unique model, by using some variant of Petri Nets, such as e.g. Hybrid Functional Petri Nets [9, 10] or Continuous Petri Nets [16] or Stochastic Petri Nets (SPNs) [8]. The last ones model the stochastic nature of a reaction system especially well. The ODE model can be obtained by means of punctual measured data and using interpolation, cf. [4], or from a first established Modular Interaction Network, cf. [18]. The SPN model, as graph models in general, can be obtained from a system of ODEs which describes the reaction system; such translations are well explained in e.g. [6,9,10]. In general, a system of ODEs defines a unique SPN but it is possible that different systems of ODEs define the same SPN. Conditions for one-to-one and onto mappings between ODEs and SPNs are given in [16].

Please note that an essential point while constructing an SPN-model is the definition of its initial marking. It should faithfully map the initial concentrations of all species involved in the reaction system.

Considering the models quoted above, it is not possible by formally analyzing it, to extract the steady situation[1] in which the reaction system may stay after some time. The only thing which can be done, and which is done in general, is to simulate the model (over a very high number of runs) until being able to deduce properties concerning the steady situation[1], with some remaining uncertainty.

In this paper we are using the uncertain data obtained by simulation of the SPN, and also uncertain parameters estimated by measures and interpolation, and prove analytically if they present in fact those of a steady situation[1]. For this reason, we first observe the SPN-model during a high number of runs which yield mean values corresponding to the concentration of species and reaction rates in the steady situation[1]. In function of these values, we map the SPN onto a Time Petri Net-model (TPN), having the same skeleton, such that the behaviour (that of the observed steady situation[1]) stays the same.

This works, because TPNs have the same semantics as SPNs with constant rates, and we dispose of a well established theory [11–13] for studying analytically their behaviour. In particular, our reduction results concerning state spaces of TPNs [12, 13] will be of good use in the presented work. When the simulated reaction rates in the steady situation[1] are exactly the rates in the real situation, then the reduced reachability graph of the TPN should consist of cycles only - up to some initiation part. By contraposition we may conclude, that a non

cyclic form of the reduced state space indicates severe problems in the set of data used to build the TPN, and by consequence, in the initially established data from experiences in the wet labs. Furthermore, we are able to calculate the time-length of the cycle(s), a data which cannot be measured within the SPN model. The last one can be compared with measures from the wet labs, if there are any. Thus once more, we bridge back to the original data. Thus our method offers a way of validating a complex modeling process of biochemical reaction systems.

This paper is organized as follows: in the next section we recall some basic notions and notations of the used Petri Net classes together with the reduction results of TPN state spaces. In section 3 we introduce the mapping from an SPN in the steady situation[1] onto a TPN model. In the subsequent section we illustrate our approach on the core model of the influence of the Raf-1 Kinase Inhibitor Protein (RKIP) on the Extracellular signal Regulated Kinase (ERK) signalling pathway, chosen as running example, before concluding.

2 Basic Concepts

In this section we recall the concept of TPNs. After that we introduce some basic notions and fundamental properties, which are important for their quantitative and qualitative evaluation.

2.1 Basics

Time Petri Nets (TPN) [11] are derived from classical Petri nets by assigning to each transition t a (continuous) time interval $[a_t, b_t]$. Here a_t and b_t are relative to the time when t was enabled most recently. When t becomes enabled, it can not fire before a_t time units have elapsed, and it has to fire not later than b_t time units, unless t got disabled in between by the firing of another transition. The firing itself of a transition does not consume time. So, the given time intervals specify reaction times for the transition firings. The time intervals are defined on non-negative real numbers, but the interval bounds are given as nonnegative rational numbers. Rational numbers are sufficient to reflect any measuring accuracy required by a given application domain. Moreover, to support the normalization of different time scales within a model, zero and ∞ are allowed as interval bounds.

As usual, in this paper, $\mathbb{N}$ denotes the set of natural numbers, and $\mathbb{Q}_0^+$, resp. $\mathbb{R}_0^+$, the sets of nonnegative rational numbers, resp. real numbers. T^* denotes the set of all finite words over the alphabet T, $l(w)$ is the length of a given word w.

Some 5-tuple $\mathcal{Z} = (P, T, v, m_o, I)$ is called a **Time Petri net (TPN)**, if $S(\mathcal{Z}) := (P, T, v, m_o)$, the **skeleton** of $\mathcal{Z}$, is a Petri net where P, T are finite sets with $P \cap T = \emptyset$, $v : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ defines the arcs with their weight, $m_o : P \longrightarrow \mathbb{N}$ fixes the initial marking, and $I : T \longrightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$ is its

interval function where $\forall t \in T$, $I(t) = [I_1(t), I_2(t)]$ and $I_1(t) \leq I_2(t)$, specifying the **earliest** and **latest firing time of** t: $eft(t) = I_1(t)$, $lft(t) = I_2(t)$.

As shown in [11], considering TPNs with $I : T \longrightarrow \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ will not result in a loss of generality. Therefore, only such time functions I will be considered subsequently.

A marking $m : P \longrightarrow \mathbb{N}$ can be seen as a vector of size $|P|$, we refer to it as p**-marking**. Thus each transition $t \in T$ induces the p-markings t^-, t^+ and Δt defined by $t^-(p) := v(p,t)$, $t^+(p) := v(t,p)$ and $\Delta t(p) := t^+(p) - t^-(p)$. With these notions the **firing rule** for TPNs can be defined. A transition $t \in T$ is **enabled** at a marking m iff $t^- \leq m$ (e.g. $t^-(p) \leq m(p)$ for every place $p \in P$).

The **pre-sets** and **post-sets** of a place or transition x are given by ${}^\bullet x := \{y \mid v(y,x) > 0\}$ and $x^\bullet := \{y \mid v(x,y) > 0\}$, respectively.

An example for an arbitrary TPN is shown in Fig. 1.

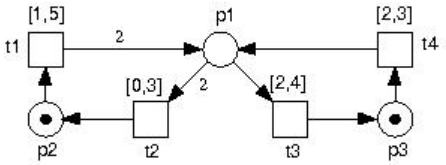


Fig. 1. A Time Petri net $\mathcal{Z}_1$.

Every possible situation in a given TPN can be described completely by a **state** $z = (m, h)$, consisting of a p-marking m (the standard marking) and a transition-marking (short: t-marking) h. The **t-marking** is a transition vector, which describes the current time circumstances in a certain situation. More exactly, each component of the t-marking is either a real number or the sign $\sharp$. Thus $h(t)$ can be seen as clock of t. If t is enabled at a marking m, its clock $h(t)$ shows the time elapsed since t became most recently enabled. If t is disabled at m, the clock is switched off (indicated by $h(t) = \#$).

Formally, a pair $z = (m, h)$ with $m : P \longrightarrow \mathbb{N}$ and $h : T \longrightarrow \mathbb{R}_0^+ \cup \{\#\}$ is called a **state** of a TPN $\mathcal{Z} = (P, T, v, m_o, I)$ if $\forall t \in T$, either ($t^- \leq m$ and $h(t) \leq lft(t)$) or ($t^- \not\leq m$ and $h(t) = \#$).

The **initial state** $z_o := (m_o, h_o)$ of the TPN $\mathcal{Z}$ is given by defining h_o as follows

$$\forall t \in T, h_o(t) := \begin{cases} 0 & \text{if } t^- \leq m_0 \\ \# & \text{if } t^- \not\leq m_0. \end{cases}$$

Thus, the initial state of $\mathcal{Z}_1$, as given in Fig. 1, is $z_0 = (\underbrace{(0,1,1)}_{p\text{-marking}}, \underbrace{(0,\sharp,\sharp,0)}_{t\text{-marking}})$.

The state $z = (m, h)$ is called an **integer state**, if $h(t)$ is an integer for each enabled transition t in m.

The behaviour of a TPN is defined by changing from one state into another by **firing a transition** (without auto-concurrency) or by **time elapsing**. In

order to define these dynamic aspects of TPNs we need first the notion **ready to fire**.

Let t_o be a transition in T and $z = (m, h)$ a state of a TPN $\mathcal{Z} = (P, T, v, m_o, I)$. The **transition** t_o is **ready to fire** at state z, denoted by $z \xrightarrow{t_o}$, if $t_o^- \leq m$ and $eft(t_o) \leq h(t_o)$.
Then the **state** z **change** into a state $z' = (m', h')$ by the **firing** of such a t_o, denoted by $z \xrightarrow{t_o} z'$, where the new marking is $m' = m + \Delta t_o$ and the new clock satisfies $\forall t \in T, h'(t) := \begin{cases} \# & \text{if } t^- \not\leq m' \\ h(t) & \text{if } t \neq t_0,\ (t^- + t_o^-) \leq m,\ t^- \leq m' \\ 0 & \text{otherwise.} \end{cases}$

This definition implies, that in the case that t_0 is still enabled after the firing of t_0, it can only refire after at least α_t new waiting units. To resume, concurrency but no auto-concurrency is possible by the way the evolving of the clocks is defined.

The **state** z may also **change** into a state $z' = (m, h')$ **by the time elapsing** $\tau \in \mathbb{R}_0^+$, denoted by $z \xrightarrow{\tau} z'$, where the marking stays the same, but time goes on : $\forall t \in T$ with $h(t) \neq \#$ we need $h(t) + \tau \leq lft(t)$) i.e. the time elapsing τ need to be possible, and the new clock is given $\forall t \in T$ by

$$h'(t) := \begin{cases} h(t) + \tau & if\ \ t^- \leq m' \\ \# & if\ \ t^- \not\leq m'. \end{cases}$$

A state $z = (m, h)$ of a TPN $\mathcal{Z}$ is called **reachable** in $\mathcal{Z}$ (starting at z_0), if there exist states $z_1, z_1', ..., z_n, z_n'$, transitions $t_1, ..., t_n$, and times $\tau_i \in \mathbb{R}_0^+$, for $i \leq n$, such that $z_0 \xrightarrow{\tau_0} z_1 \xrightarrow{t_1} z_1' \xrightarrow{\tau_1} z_2 \xrightarrow{t_2} z_2' \xrightarrow{\tau_2} \dots z_n \xrightarrow{t_n} z_n' \xrightarrow{\tau_n} z$ holds.

The **sequence of transitions** $\sigma = t_1 \dots t_n$ leading to a reachable state will be called a **feasible** one (starting at z_0) or just a **firing sequence** of $\mathcal{Z}$. The full sequence $\sigma(\tau) = \tau_0 t_1 \tau_1 \dots t_n \tau_n$ is called a **(feasible) run** of σ. It shows that in a given TPN the state changes generally consist of alternating series of time elapsing and transition firing. Obviously, for a given run the transition sequence is well defined, and for a given firing sequence there are infinitely many runs in general.

Eventually, $RS_{\mathcal{Z}}(z')$ is the **set of all reachable states** in $\mathcal{Z}$ starting from an arbitrary state z'. And $RS_{\mathcal{Z}} := RS_{\mathcal{Z}}(z_0)$, that from the initial state is also called the **state space** of $\mathcal{Z}$.

We may also consider the **set of reachable** p**-markings**, also called the $p-$**marking space** $R_{\mathcal{Z}} := \{\ m \mid (m, h) \in RS_{\mathcal{Z}}\}$ in a TPN $\mathcal{Z}$. This is a subset (not necessarily proper) of the reachable markings of the skeleton $S(\mathcal{Z})$. Therefore, a firing sequence in the skeleton $S(\mathcal{Z})$ is *not necessarily* a firing sequence in $\mathcal{Z}$. The set of p-markings, reachable in $\mathcal{Z}$ starting at an arbitrary p-marking m', is denoted by $R_{\mathcal{Z}}(m')$. A TPN is called **bounded**, if its set of reachable p-markings is finite, otherwise it is called **unbounded**.

For different reasons the state space of a TPN is in general infinite and dense in terms of the time: the set of reachable p-markings can be infinite or the set of t-markings for a fixed p-marking can be infinite or both together. Later on, we

consider some approaches for concise state space representations, when $RS_{\mathcal{Z}}$ is infinite while $R_{\mathcal{Z}}$ is finite.

The definition of state change by time elapsing can be slightly and consistently modified for the introduction of a reachability graph based on all **reachable essential states** for arbitrary TPNs, especially for TPNs including transitions whose *lft*s are ∞. The set of all **reachable essential states** for arbitrary TPNs is defined as a subset of all reachable integer states of the considered TPN. We will use the following property (for more details see [13]): if no transition t in $\mathcal{Z}$ has $lft(t) = \infty$ then the set of essential states is exactly the set of integer states in $RS_{\mathcal{Z}}$.

2.2 Time-dependent Minimal and Maximal Runs in TPNs

We will formalize the time-dependent notions of measuring the length of runs, cf. [15].

Let $\sigma(\tau)$ be a run of the transition sequence σ in some TPN $\mathcal{Z}$. The **length of the run** $l\big(\sigma(\tau)\big)$ is the sum of all times while executing the run $\sigma(\tau)$, i.e.,

$$l\big(\sigma(\tau)\big) := \sum_{i=0}^{n} \tau_i, \quad \text{where} \quad n = l(\sigma) \quad \text{and} \quad \tau = \tau_0\tau_1 \ldots \tau_n.$$

For a given transition sequence σ in $\mathcal{Z}$, a feasible run $\sigma(\tau)$ with minimal length will be referred to as **minimal run** of σ. Evidently, it satisfies :

$$l\big(\sigma(\tau)\big) := \min_{\tau'}\{\, l\big(\sigma(\tau')\big) \mid \sigma(\tau') \text{ is a feasible run of } \sigma \text{ in } \mathcal{Z}\,\}.$$

The notion of **maximal run** can be introduced analogously. It denotes the run with maximal length within all feasible runs of σ if such an upper bound exists; otherwise it is not defined.

The notions of **minimal**, respectively **maximal time distance between two states** can be found in [12, 15] and are useful for precise analysis of biochemical reaction systems which present different kind of steady situations[1] than our current running example.

2.3 State Space Reduction

The central problem for the dynamic analysis of a given TPN is the adequate knowledge of its state space. It is important to get a finite description of the infinite state spaces, under the condition that the p-marking space is finite.

It can be shown that - despite the continuous nature of the time intervals - it is sufficient to pick up just some "essential" states to determine the entire

timed behaviour of the net so that qualitative and quantitative analyses remain possible.

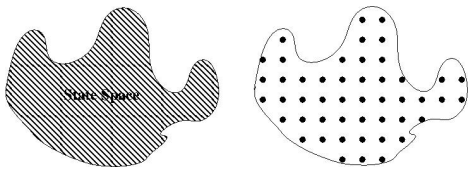


Fig. 2. For some TPN $\mathcal{Z}$. Left hand. Sketched state space of $\mathcal{Z}$: a continuous set. Right hand. Sketched reduced state space of $\mathcal{Z}$: all reachable integer states.

While the calculation of a single reachable integer state is rather straightforward, the proof that *the knowledge of the integer states is sufficient for analyzing a TPN* was quite difficult. Three solutions had been proposed in the past: considering a global clock [11] or considering a parametrical description of the state space [15] or dividing into a finite number of problems, which can be solved recursively with a methodology inspired from dynamic programming [12]. As result, one is able to construct for each TPN a **reduced reachability graph** whose vertices are the **essential states**. When the TPN does not contain a transition whose lft is ∞, then the essential states are exactly all the reachable integer states in the net.

An edge (z_1, z_2) labeled (k, t), with $k \in \mathbb{N}$, in this reduced reachability graph has the meaning that in state z_1 k time units are elapsing before transition t fires leading to state z_2. Now, for finding minimal and maximal time paths between two states/p-markings in a TPN, its reduced reachability graph can be used, even effectively. Our algorithms for computing the reduced reachability graph of a given TPN are implemented in several standard Petri Net tools, like INA [17], tina [3] and charlie [7]. INA can additionally compute minimal and maximal time-dependent paths. Thus these tools can be successfully applied for models of bio-chemical reaction systems, too.

2.4 Stochastic Petri nets

Stochastic Petri Nets (SPNs) had been introduced at the beginning of the Eighties, cf. [1, 2]. They are widely used in the modeling of biochemical reaction systems, cf. [8].

Such SPNs are derived from classical PNs by assigning to each transition t a firing rate λ_t. This firing rate specifies a firing delay for the transition. More exactly, the firing delay is a random variable which is distributed exponentially

and has λ_t as parameter of the probability density function. In fact, to each transition t a probability density function with parameter λ_t is associated :

$$f_t(x;\lambda_t) = \begin{cases} \lambda_t e^{-\lambda_t x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

Finally, the firing rate λ_t may be marking-dependent in general. In such a case, we should write $\lambda_t(m)$, where m is a marking, instead of λ_t. Than the expected value for the firing delay for the transition t in the marking m is $\frac{1}{\lambda_t(m)}$.

The firing mode is defined as follows: In a given marking m, each enabled transition t obtains an instance of the firing delay $\lambda_t(m)$ from its associated probability density function. Then a choice is made: the transition with the minimum firing delay is firing. The firing itself of a transition does not consume time. The successor marking is than obtained as in the underlying classical PN. It is well known [1], that the probability for two transitions to fire at the same instant is null, i.e. there is no conflict. That is why the transitions in SPNs fire naturally one by one, i.e., just as in TPNs.

3 Biochemical Systems and Time Petri Nets

Biochemical reaction networks are mostly described by ordinary differential equations (ODEs) or reaction rate equations (RREs), and both can be converted into each other. Taking account of the rate equations of all reactions in the systems, ODE like RRE models can be transformed into Continuous Petri nets or Stochastic Petri nets. More about these transformations can be found, e.g. in [8]. Conditions for a uniform transformation of ODEs into Continuous Petri nets (or RREs) are introduced in [16]. Systems of ODEs can be represented as hybrid functional Petri nets, cf. [9, 10], too. These Petri net models allow for qualitative and quantitative evaluations using tools and methods of the Petri net theory, cf. [2, 5].

A transformation of an ODE model into a Time Petri net model (TPN) using the reaction rates is shown in [14]. This transformation allows the computation of time-minimal and time-maximal paths (if existing) between two system situations, i.e. two states of the TPN model. It can be considered as an indication for the conformance and coherence of the model if the length of the time-minimal and time-maximal paths coincide with the results in the wet labs. Otherwise the original model becomes invalidated.

Independently from the original model, an RRE one or an ODE one, in a first step, a timeless Petri net is always derived. This describes the causal relations between the events in the system. In biochemical systems, these are biochemical reactions or biochemical signal transductions. Thereafter additional information, in particular the time parameters, need to be assigned to the Petri net. They are obtained from the parameters (kinetic rate constants) in the ODEs. Their values are often determined experimentally. When it is not possible to collect

or identify them in vitro, the parameters are estimated using experimental data achieved only for some discrete time points. In this case the goal is to estimate the value of the parameters for each moment so that the values over the time fit the experimental data (cf. [9]). Thus in a second step, integrating these parameters, a time-dependent PN model is established for the biochemical network. It is obviously that at this stage of modeling a certain level of inexactness is present in each model.

In this paper we are going to study biochemical reaction systems which possess a steady situation[1]. This is the case when the system comes in a situation, in which the concentration of all substances stays constant. Usually, in the steady situation[1] the concentration of all substances allows that all reactions take place permanently. Now constructing the reachability graph, may be interpreted as considering the path of changes of the single substances. Loosely speaking, we should get a cyclic set of states in the reachability graph corresponding to the steady situation[1]. The behaviour of the SPN, expressed by Markow chains is isomorphic to the full reachability graph of the underlying PN, i.e., they have the same state space. By convention, we speak in the following of "the reachability graph" of the SPN. But the nodes corresponding to the steady situation[1] **can not** be recognized in this reachability graph, even knowing the rates in the steady situation[1]. To our knowledge, no method is known until now for separating or extracting the subgraph corresponding to the steady situation[1] from the reachability graph of the SPN. Steady state meaning cyclic behaviour, this subgraph (of the reachability graph of the SPN) is supposed to present a cyclic structure (with one or more circles), up to some initiation part. In contrast, knowing the steady situation[1] rates we are able to separate the reachable states, we are looking for, using a TPN and its reachability graph. This is due to the reduction results on reachability graphs of TPNs discussed in section 2.3.

Thus, we propose in this paper a methodologie to calculate and verify such set of states which correspond to steady situation[1]. The starting point will always be an SPN model for a biochemical reaction system, which has a steady situation[1]. This means that the rate for each enabled transition in each marking is constant. The steady situation[1] concentrations and rates can be determined using simulation of the SPN. Examples for which about 10,0000 simulation runs have been done may be considered. These runs has to be merged into one averaged simulation run showing the mean of the concentrations, and thus also of the rates, over the time. We take the expectation values of the steady situation[1] rates for our investigations.

Simulation means approximation; thus it is not a priori clear how accurate the determined steady situation[1] rates are.

The reciprocal value of the rate is the time which each enabled transition has to wait before it can fire. The transition with the minimal waiting time fires in an SPN. Consequently, an SPN acts in the steady situation[1] exactly like a certain kind of TPN: we propose to construct a TPN, having the same underlying Petri net as the SPN, and where the transitions t will recieve time intervals $[a_t, b_t]$,

where $a_t = b_t$ is equal to the above calculated waiting time of t in the SPN in the steady-state.

A qualitative analysis of the TPN can prove whether the subset of all reachable states generate cycle(s) only (up to some initiation part). Furthermore, the time length of these cycles can be computed.

The formal analysis we proposed allows the following interpretations. If the reduced reachability graph of the TPN consists of cycles, then the considered rates achieved by simulation describe a steady situation[1], actually. By contraposition we may deduce, that a non cyclic form of the reduced state space indicates severe problems in the set of data used to build the TPN, and by consequence, in the initially established data from experiences in the wet labs.

Additionally the time-length of the cycles can be easily computed and compared with results from the wet labs. Both, the reachability graph of the TPN and the time-length of the cycles are either an indication for the correctness of the models or they invalidate these. Therefore our method offers a way of validating a complex modeling process of biochemical reaction systems.

4 An Example

In this section we will illustrate our approach of analysing a biochemical reaction system in a steady situation[1] along an example introduced in [4] and studied further in [6, 8], concerning the core model of the influence of the Raf-1 Kinase Inhibitor Protein (RKIP) on the Extracellular signal Regulated Kinase (ERK) signalling pathway.

In [4] this biochemical reaction system is modeled using an integrated approach of mathematical modeling in combination with experimental data. This model consists of eleven nonlinear ODEs. The parameters in the ODEs are estimated using interpolation of polynomial functions.

Afterwards, simulation studies provides a qualitative validation of the mathematical model compared to experimental results in the wet labs in view of the transient behavior and sensitivity analysis. However, parameter estimation is, as already mentioned, an uncertain factor in such a mathematical model.

Then in [6, 8], a qualitative model is proposed in terms of a Petri Net, see Fig. 3, deduced from the quoted ODE system.

Additionally, reaction rates are associated to all transitions of this PN [6, 8]. These are derived from the estimated parameters used in [4]. The obtained whole model is therefore an SPN. In [6], inter alia, the example is considered w.r.t. the attained steady situation[1] in the biochemical network. This is done by simulation: Rate values for the reactions are estimated after about 10.000 simulation runs have been done. Nevertheless, considering the behaviour of the model based on estimated parameters, we are in presence of a further factor of uncertainty.

We are going to investigate the SPN in the simulated steady situation[1]. First let us have a look on its reachabilty graph depicted on the left hand side of Fig.4. Unfortunately no method exists, to our knowledge, to find out analytically

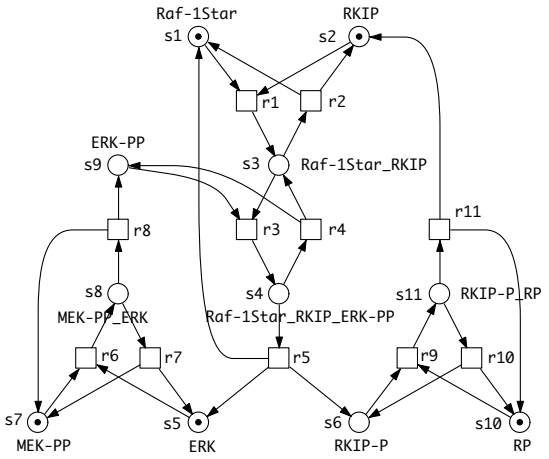


Fig. 3. The Petri net for the core model of the RKIP pathway, consisting of 11 places and 11 transitions. The places s1, ..., s11 stand for proteins or protein complexes. Complexes are indicated by an underscore _ between the protein names, phosphorylated forms by the suffix -P or -PP. The transitions r1, ..., r11 model the reactions. The preplaces of a transition correspond to the reaction's precursors, and its postplaces to the reaction's products. The layout follows the suggestions by the graphical notation used in [4]. The initial marking is constructed systematically using standard Petri net analysis techniques. *This figure with its legend is cited from [8].*

/ formally which are the states (nodes) corresponding to the steady situation[1]. Instead, we will use the estimated data in order to derive a TPN. This time-dependent Petri net should have the same state space as the SPN in the steady situation[1]. To be able to do this derivation, we need to know the waiting (or delay) times τ_i. They can be calculated from three kind of informations/data, given in the tables below.

- the **rate function** v_i, presented in [8] for each of the eleven transitions r_i, and shown in Table 1
- the estimated parameters $k_1 \cdots k_{11}$ in the eleven corresponding ODEs, presented in [4], renamed **rate parameters** and denoted by $c_i := k_i$ in [8], and shown in Table 1
- the **mean steady situation[1] concentrations** for the species s1 $\cdots$ s11 are taken from [6] and shown in Table 2.

Table 1. The rate function for each transition and the rate constants (the estimated parameter in the ODEs), *cited from [4, 8]*. The abbreviations s1 $\cdots$ s11 stand for the involved species as follows: s1 is Raf-1*, s2 is RKIP, s3 is Raf-1*_RKIP, s4 is Raf-1*_RKIP_ERK-PP, s5 is ERK, s6 is RKIP-P, s7 is MEK-PP, s8 is MEK-PP_ERK, s9 is ERK-PP, s10 is RP and s11 is RKIP-P_RP. In the rate functions each of the s1 $\cdots$ s11 is supposed to be the mean concentration of the species s1 $\cdots$ s11 in the simulated steady situation in the SPN, as given in Table 2.

transition r_i	rate function v_i	rate constant c_i
$r1$	$c1 \cdot s1 \cdot s2$	0.053
$r2$	$c2 \cdot s3$	0.0072
$r3$	$c3 \cdot s3 \cdot s9$	0.625
$r4$	$c4 \cdot s4$	0.00245
$r5$	$c5 \cdot s4$	0.0315
$r6$	$c6 \cdot s5 \cdot s7$	0.8
$r7$	$c7 \cdot s8$	0.0075
$r8$	$c8 \cdot s8$	0.071
$r9$	$c9 \cdot s6 \cdot s10$	0.92
$r10$	$c10 \cdot s11$	0.00122
$r11$	$c11 \cdot s11$	0.87

Table 2. Mean steady situation[1] concentrations for for all s_i, *cited from [6]*.

specie s_i	concentration
$s1$	0.2133
$s2$	0.1727
$s3$	0.2163
$s4$	0.5704
$s5$	0.0332
$s6$	0.0200
$s7$	0.7469
$s8$	0.2531
$s9$	0.1433
$s10$	0.9793
$s11$	0.0207

Now, we can calculate the rates $v_1 \cdots v_{11}$ of the transitions in the steady situation[1] using their rate functions from Table 1 and the data from Table 1 and Table 2. Subsequently, the delay time τ_i for every one of the ten transitions r_i is obtained as the reciprocal of the rate in the steady situation[1]. The resulting values are presented in Table 3.

Finally, the TPN model can be constructed: As skeleton of the TPN model we take the underlying PN of the SPN, i.e. the net given in Fig. 3. To each

Table 3. Rates in the steady situation and delay times for r1 $\cdots$ r11.

transition r_i	rate in the steady state v_i	delay time in the steady state τ_i (rounded)
$r1$	0.00195235623	512
$r2$	0.00155736	642
$r3$	0.019372369	52
$r4$	0.00139748	716
$r5$	0.0179676	56
$r6$	0.019837664	50
$r7$	0.00189825	527
$r8$	0.0179701	56
$r9$	0.01801912	55
$r10$	0.000025254	39598
$r11$	0.018009	56

transition $r_i,\ 1 \leq i \leq 11$, a time interval $[\tau_i, \tau_i]$ is associated, where τ_i is the calculated delay time, from Table 3.

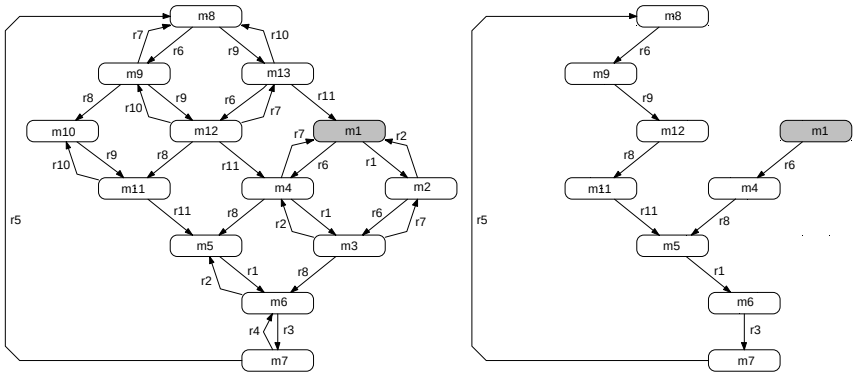


Fig. 4. Left hand: The reachability graph, from [8], for the SPN of Fig. 3.
Right hand: The reachable p-markings in the TPN. Please note, that this is not the reachability graph of the TPN.

Now the obtained TPN may be analysed. First, we just calculate the reachable p-markings, designed on the right hand side of Fig.4. We observe that due to the time constraints this graph has 9 nodes, i.e., much less p-markings are reachable as in the reachability graph of the SPN, depicted on the left hand side, which is also the reachability graph of the underlying net of the SPN and TPN. We also detect that the right graph is clearly a subgraph of the left one. The complete state space of the TPN is -a priori- infinite, a lot of states may share the same p-marking.

The reduced reachability graph of the considered TPN can now be constructed, by applying the reduction method described in section 2.3. We did it with tools INA and Charlie [7, 17], which gave us the same result, depicted in Fig 5. It consists of eleven essential states, i.e., 10 pairs of p- and t-markings, although only 10 p-markings are reachable in the considered TPN. This is no incoherence : Two essential states, $z3$ and $z10$, share the same p-marking $m5$. However, in the cycle each p-marking belongs to exactly one state (node) only.

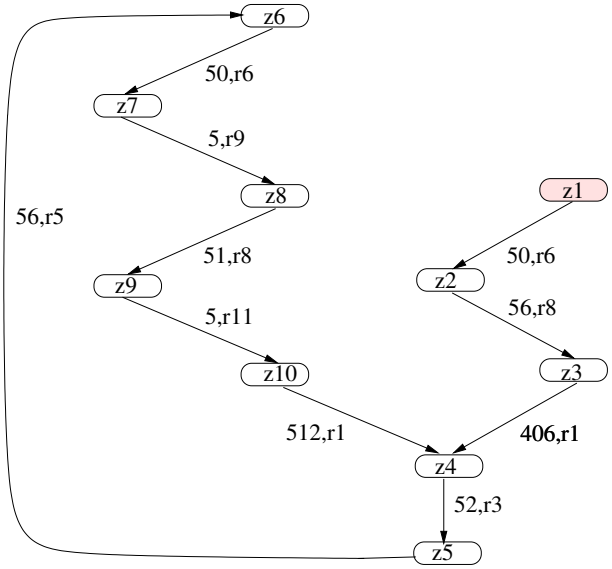


Fig. 5. The reduced reachability graph of the TPN model.

Analyzing this reduced reachability graph of the TPN tells us that it consists of the **cycle** $z4, r3, z5, r5, z6, r6, z7, r9, z8, r8, z9, r11, z10, r1$ and an **initiation path** $z1, r6, z2, r8, z3, r1$.

This path (or panhandle) is caused by the choice of the initial p-marking for the TPN, chosen to be the same as for the SPN. Actually, the initial p-marking for the TPN should be a p-marking which the SPN reaches in the steady situation[1]. However, the TPN only initiates its behaviour by this path and then comes to the steady situation[1], i.e. stays in the cycle. The time-length of the cycle was also calculated, its value is 731 time units.

We also read on this reduced graph that the transitions $r2, r4, r7$ and $r10$ will never fire. Such transition are called *dead*. These are the transitions modeling the backward reactions which have rate constants being essentially smaller as the

rate constants for the forward reactions. This tell us that in the steady situation[1] the backward reactions do never proceed.

5 Conclusions

In this paper we introduce a method for qualitative and quantitative evaluation of an SPN model of biochemical reaction systems in a steady situation[1] including validation of all used data. A mathematical model of such a system contains a number of uncertain factors resulting from the estimation of the parameters in the ODEs and the values of the reaction rates in the steady situation[1] obtained by simulation of the SPN. The reaction rates and the concentration of the species in the steady situation[1] are constant values.

This means that the set of reachable markings in the SPN model in the steady situation[1] is finite and they generate a cycle, not necessarily a simple one. But no state reachability analysis of the SPN does allow for isolating those states which correspond to the steady situation[1], i.e. does allow to detect the cycle.

Due to the fact of constant values, we are able to propose a mapping from the usual SPN model in the steady situation[1] onto a TPN model which has the same behaviour. Contrarily to the SPN model, we can reduce the state space of the TPN to the part we are interested in, consisting of the essential states. The obtained reduced state graph can be further analyzed. Its cyclic or non cyclic form validate or invalidate the used data during the modeling process. The time length of the cycle can be calculated and compared to real time measures, too.

The algorithms for reachability analysis of TPNs, implemented in the tools [3,7,17] had been applied for the evaluation of the simulated steady situation[1] in a mathematical model of our running example. We considered the core model of the influence of the Raf-1 Kinase Inhibitor Protein (RKIP) on the Extracellular signal Regulated Kinase (ERK) signalling pathway. We were able to show that the simulated values for the reaction rates define one cycle in the TPN model and to compute the time-length of this cycle. Furthermore we ascertain that the backward reactions do not proceed in the steady situation[1].

We will lead some reflexions if the initial state for the TPN could be redefined in a better way by regarding the values for the concentrations of the species in the simulated steady situation[1]. We are planning to apply the presented method to some other cases of biological or biochemical interaction networks, were more complex steady situations[1], with -a priori- non simple cycles.

References

1. M. Ajmone Marsan. Stochastic Petri nets: An elementary introduction. In *In Advances in Petri Nets*, pages 1–29. Springer, 1989.
2. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.
3. B. Berthomieu. *TIme petri Net Analyzer*. LAAS / CNRS, 7, avenue du Colonel Roche, 31077 Toulouse, France, http://www.laas.fr/bernard/tina/, 2.9.8 released edition, 2009.

4. K.-H. Cho, S.-Y. Shin, H.-W. Kim, O. Wolkenhauer, B. McFerran, and W. Kolch. Mathematical modeling of the influence of RKIP on the ERK signaling pathway. *Lecture Notes in Computer Science*, 2602:127–141, 2003.
5. R. David and H. Alla. *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*. Prentice Hall, New York London Toronto Sydney Tokyo Singapure, 1992.
6. D. Gilbert and M. Heiner. From Petri Nets to Differential Equations - an Integrative Approach for Biochemical Network Analysis. In *Proc. ICATPN 2006, Turku, June, Springer LNCS 4024*, pages 181–200, 2006.
7. M. Heiner. *Charlie*. Brandenburgische Technische Universität, http://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Charlie, January 2011.
8. M. Heiner, R. Donaldson, and D. Gilbert. *Petri Nets for Systems Biology*, chapter 3, pages 61–97. Jones & Bartlett Learning, LCC, 2010.
9. G. Koh, D. Hsu, and P.S. Thiagarajan. Incremental Signaling Pathway Modeling by Data Integration. In *Proc. of the 14th International Conference on Research in Computational Molecular Biology (RECOMB 2010), Lisabon, Portugal*, 2010.
10. H. Matsuno, Y. Tanaka, H. Aoshima, A. Doi, M. Matsui, and S. Miyano. Biopathways representation and simulation on hybrid functional Petri net. *Silico Biology*, 3(3):2592–2601, 2003.
11. L. Popova. On Time Petri Nets. *J. Inform. Process. Cybern. EIK 27(1991)4*, pages 227–244, 1991.
12. L. Popova-Zeugmann. Time Petri Nets State Space Reduction Using Dynamic Programming. *Journal of Control and Cybernetics*, 35(3):721–748, 2007.
13. L. Popova-Zeugmann. *Time and Petri Nets (in German)*. Habilitation Thesis, Humboldt Universität zu Berlin, 2007.
14. L. Popova-Zeugmann. Quantitative evaluation of time-dependent Petri nets and applications to biochemical networks. *Natural Computing, Springer Netherlands*, pages 1–27, 2010.
15. L. Popova-Zeugmann and D. Schlatter. Analyzing Path in Time Petri Nets. *Fundamenta Informaticae (FI) 37, IOS Press, Amsterdam*, pages 311–327, 1999.
16. S. Soliman and H. Heiner. A Unique Transformation from Ordinary Differential Equations to Reaction Networks. *PLoS ONE 5(12): e14284*, 2010.
17. H.-P. Starke. *INA – The Intergrated Net Analyser*. Humboldt Universität zu Berlin, http://www2.informatik.hu-berlin.de/∼ starke/ina.html, 2003.
18. A. Yartseva, R. Devillers, H. Klaudel, and F. Képès. From MIN model to ordinary differential equations. *J. Integrative Bioinformatics*, 4(3), 2007.

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.87-101

Two modeling methods for signaling pathways with multiple signals using UPPAAL

Shota Nakano and Shingo Yamaguchi

Graduate School of Science and Engineering, Yamaguchi University
2-16-1 Tokiwadai, Ube, 755-8611, Japan
E-mail : {r033vk, shingo}@yamaguchi-u.ac.jp

Abstract. Model checking has been attracting attention to analyze signaling pathways. There are two or more, i.e. multiple, signals flowing in a signaling pathway. Most of previous work, however, have treated only one, i.e. single, signal. To analyze a signaling pathway more precisely, it is necessary to treat multiple signals. There is few previous work treating multiple signals. It is known that a primary issue in model checking is the state-space explosion. Multiple signals make it difficult to analyze by model checking. In this paper, we propose two modeling methods for signaling pathways with multiple signals. These methods transform a Petri net model of a signaling pathway to an automaton model of UPPAAL. The first method uses multiple automata as a model of UPPAAL. The second method uses a single automaton as a model of UPPAAL. We apply these methods to an example. And we find that the single automaton modeling method is more effective than the multiple automata modeling method from the viewpoint of the number of signals, the number of states explored, and checking time. These results show that the model size to be analyzed is improved by devising of modeling method.

1 Introduction

Signaling pathway is a signaling mechanism to unify the behavior of cells. Since a pathway is large and complex, there are unknown mechanisms and components. Some researchers have applied model checking to analysis of signaling pathways. Model checking is an automatic and usually quite fast verification technique for finite state concurrent systems. In 2003, Chabrier et al.[1] applied NuSMV to analysis of biological pathways, and in 2009, Bos et al.[2] applied UPPAAL to analysis of a signaling pathway. They also described that model checking is powerful, but the state-space explosion may happen. Note that both [1] and [2] treated only one signal.

There are two or more, i.e. multiple, signals in a signaling pathway because a ligand joins a receptor repeatedly. To analyze the signaling pathway more precisely, it is necessary to treat multiple signals. There is few previous work treating multiple signals. Kwiatkowska et al.[3] described that model checking of multiple signals cause the state-space explosion further than that of a single signal in performing model checking by PRISM.

In this paper, we propose two modeling methods for signaling pathways with multiple signals. These methods transform a Petri net model of a signaling pathway to an automaton model of UPPAAL. Cassez et al.[4] proposed a method for transforming a time Petri net to an timed automaton. But this method tends to increase the size of an automaton. Our methods can give a smaller automaton model. The first method uses multiple automata as a model of UPPAAL. The second method uses a single automaton as a model of UPPAAL. We apply these methods to an example. Then we evaluate the methods from the viewpoint of the number of signals, the number of states explored, and checking time. Finally, we discuss whether we can increase the model size to be analyzed by devising of modeling method.

In Sect.2, we present a Petri net model of signaling pathways and an automaton model of UPPAAL. In Sect.3, we propose two modeling method for a signaling pathway with multiple signals. In Sect.4, we apply the two proposed methods to an example and evaluate the two methods. Finally, Sect.5 gives a conclusion and some future work.

2 Preliminary

2.1 Petri net model

Matsuno et al.[5, 6] have proposed a Petri net model of signaling pathways. The Petri net model can represent multiple signals as multiple tokens. Places denote static elements including chemical compounds, conditions, states, substances and cellular organelles. Tokens indicate the presence of these elements. The number of tokens is given to represent the amount of chemical substances. Transitions denote active elements including chemical reactions, events, actions, conversions and catalyzed reactions. Directed arcs connecting the places and the transitions represent the relations between corresponding static elements and active elements.

The formal definition of a Petri net model of a signaling pathway is as follows.

Definition 1. *A Petri net model of a signaling pathway is a 5-tuple* $TPNR = (T, P, \mathcal{E}, D, R)$.

T : *A set of transitions* $\{t_1, t_2, \cdots, t_{|T|}\}$
P : *A set of places* $\{p_1, p_2, \cdots, p_{|P|}\}$
$\mathcal{E}$: *A set of directed arcs between the places and the transitions*
$D : T \to \mathbb{N}$: *A function to assign a firing delay time to a transition*
$R : T \to [0, 1]$: *A function to assign a firing rate to a transition.*
Note that $\sum_{t \in p^\bullet} R(t) = 1$. □

There are one or more source transitions and one or more sink transitions. Every node is on a path from a source transition to a sink transition. If a firing of a transition t_i is decided, tokens required for the firing are reserved. We call these tokens as reserved tokens. When $D(t_i)$ passes, t_i fires to remove the reserved tokens from each input place of t_i and put non-reserved tokens into each output place of t_i. If a place p_i has two or more output transitions, each output

transition can't fire over $R(t_i)$. Let $X(t)$ be the firing count of t. Then, $\forall t \in p^\bullet$: $\frac{X(t)}{\sum_{t' \in p^\bullet} X(t')} \leq R(t)$. Figure 1 is a Petri net model of a part of IL-1 signaling pathway[6].

2.2 Automaton model of UPPAAL

UPPAAL[7] is a model checking tool for verification of real-time systems. This tool can use timed automata as state transition models. We use the following notations: C is a set of clocks and $B(C)$ is the set of conjunctions over simple conditions of the form $x \bowtie c$ or $x - y \bowtie c$, where $x, y \in C$, $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. A timed automaton is a finite directed graph annotated with conditions over and resets of non-negative real valued clocks.

Definition 2. *A timed automaton is a 6-tuple* (L, l_0, C, Act, E, I).
L : *A set of locations*
$l_0 \in L$: *The initial location*
C : *A set of clocks*
Act : *A set of actions*
$E \subseteq L \times Act \times B(C) \times 2^C \times L$: *A set of edges between locations with an action, a guard and a set of clocks to be reset*
$I : L \to B(C)$: *A function to assign to an invariant to a location* □

Timed automata often form a network over a common set of clocks and actions, consisting of n timed automata $A_i = (L_i, l_i^0, C, Act, E_i, I_i)$, $1 \leq i \leq n$. For the semantics of the automaton model, refer to [8].

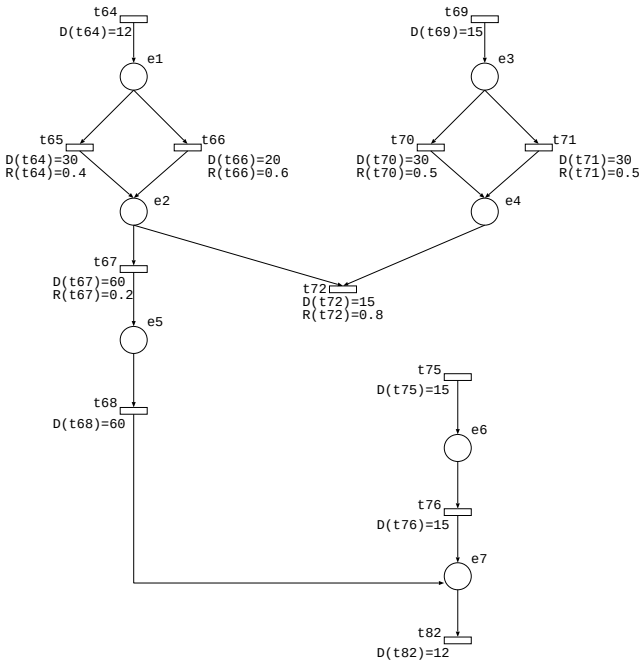


Fig. 1. A part of Petri net model of IL-1 signaling pathway

In model checking in UPPAAL, we describe a property to be analyzed in Timed Computation Tree Logic (TCTL) and we can verify whether the model with clock variables satisfies the property by running model checking. TCTL on UPPAAL includes `E<> p` (There exists a path where `p` eventually holds) and `A[] p` (For all paths `p` always holds).

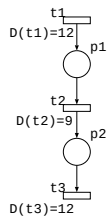
3 Two modeling methods for signaling pathways with multiple signals

A Petri net model representing a signaling pathway must be correct. We use UPPAAL to verify whether a Petri net model is correct, because the Petri net model includes time concept. The model used in this paper is timed Petri net, but it is easy to extend our method to time Petri nets.

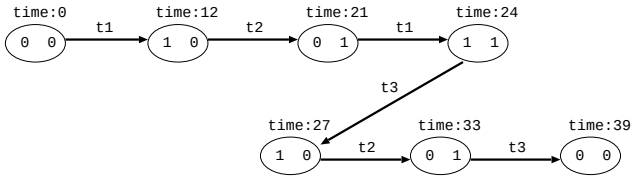
It is known that model checking is powerful but may cause the state-space explosion. The more signals are, the more the problem becomes severe. Kwiatkowska et al.[3] mentioned that model checking with multiple signals cause the state-space explosion further than that with a single signal. It is known that there is a tradeoff between expressive power and analytical power. To balance the powers, we need to devise modeling methods. Concretely, we propose two modeling methods. The first method uses multiple automata as a model of UPPAAL. It is named "multiple automata modeling method." The second method uses a single automaton as a model of UPPAAL. It is named "single automaton modeling method."

3.1 Multiple automata model and single automaton model

We give the representation model used in each modeling method. Both models have the same expressive power as a Petri net model; provided that the Petri net model has no transition branches. Note that those models can treat transition joins. We give a simple example to show the difference between the multiple automata model and the single automaton model.



(a) A Petri net model



(b) A state transition of the case t_1 fires twice

Fig. 2. A Petri net model $TPNR_1$ of a signaling pathway

(1) Multiple automata model

The multiple automata model preserves the structure of a given Petri net model as much as possible. A Petri net model with n signals is represented as the following:

- The multiple automata model of the Petri net model consists of n automata.
- Each automaton represents the state transition of a single signal.
- A place or a transition is represented as a location.
- An arc is represented as an edge.
- Firing delay time is implemented by using a location invariant and a guard.

In this example, we use a Petri net model, $TPNR_1$, which is shown in Fig.2. Figure 2(a) is a Petri net model and Fig.2(b) is a state transition of the case t_1 fires twice. Since $TPNR_1$ is state machine, the reachability graph has a similar structure as the net. Figure 3 shows the multiple automata model of $TPNR_1$. In this example, we assume that source transition t_1 fires twice. There are two signals in this model. The two signals are represented as two automata. Two places and three transitions are represented as five locations. Four arcs are represented as four edges. Firing delay time $D(t_1) = 12$ is implemented by using location invariant (`gc1<=12`) of `t1` and guard (`gc1>=12`) of edge `(t1,p1)`, where `gc1` denotes a global clock. Similarly, firing delay time $D(t_2) = 9$ is implemented by using location invariant (`c<=9`) of

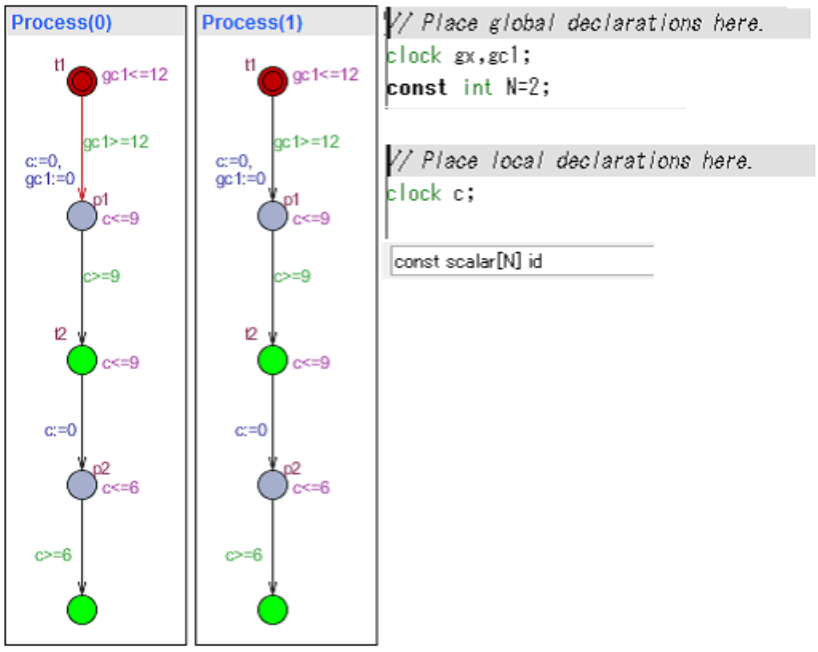


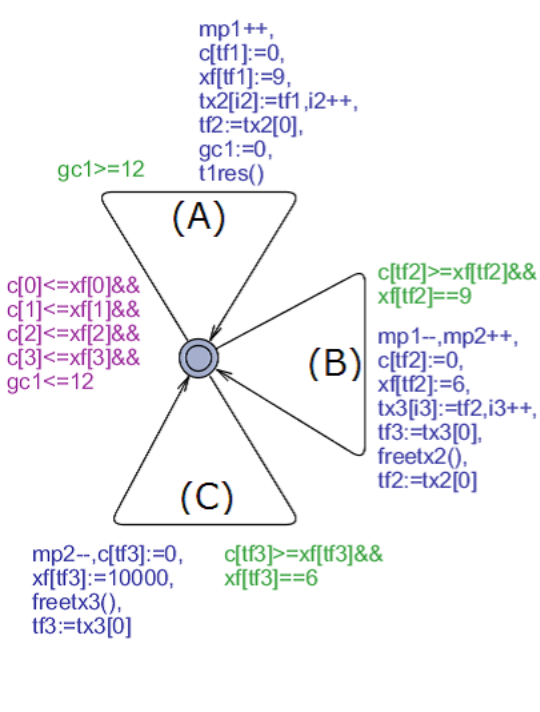
Fig. 3. The multiple automata model of $TPNR_1$

Table 1. Variables of single automaton model

Variable	Meaning
clock `gc1`, `gc2`, $\cdots$, `gc`m	Clock for source transitions
clock `c[`k`]`	Clock of the i-th signal
int `xf[`k`]`	Firing delay time of the i-th signal
int `mp1`, `mp2`, $\cdots$, `mp`$\|P\|$	The number of signals in p_i
int `tf1`, `tf2`, $\cdots$, `tf`$\|T\|$	Signal ID of the first reserved signal in transition t_i
int `tx1[`k`]`, `tx2[`k`]`, $\cdots$, `tx`$\|T\|$`[`k`]`	Signal IDs reserved in transition t_i
int `l1`, `l2`, $\cdots$, `r1`, `r2`, $\cdots$	The firing count of a transition in a place branch
void `t1res()`, `t2res()`,$\cdots$	Reset `c[]` and `xf[]`
void `freetx1()`, `freetx2()`,$\cdots$	Organize reserved signal IDs of transition t_i

`p1` and `t2` and guard (`c>=9`) of edge `(p1,t2)`, where `c` denotes a local clock. $D(t_3)$ is implemented by a similar way. A current state of the automata represents where the signals are now.

(2) Single automaton model
The single automaton model represents a given Petri net model as a single automaton with only one location even if there are two or more signals. A Petri net model with n signals is represented as the following:




```
// Place global declarations here.
clock gx,gc1;
clock c[4];
int xf[4]={2000,10000,10000,10000};
int tx2[3];
int tx3[3];
int i2,i3;
int tf1=1,tf2,tf3;
int mp1,mp2;

void t1res(){
        if(tf1==3)      tf1=1;
        else            tf1++;
}
void freetx2(){
        int j;
        for(j=1;j<=i2;j++){
                tx2[j-1]:=tx2[j];
        }       i2--;
}
void freetx3(){
        int j;
        for(j=1;j<=i3;j++){
                tx3[j-1]:=tx3[j];
        }       i3--;
}
```

Fig. 4. The single automaton model of $TPNR_1$

- The single automaton model consists of a single automaton with only one location.
- The automaton represents multiple signals by using clocks `c[]` and variables `xf[]`.
- The markings of the places are represented as variables `mp1`, `mp2`, $\cdots$, `mp`$|P|$.
- A firing of each transition is represented as a self loop of the location.
- Firing delay time is implemented by using a location invariant and a guard.

Figure 4 shows the single automaton model of $TPNR_1$. Figure 5 is the state transition of the single automaton model. Places p_1, p_2 are represented as a single location with variables `mp1`, `mp2`, where `mp`i denotes the marking of p_i. The state transition of marking is same as Fig.2(b). Table 1 is the variables used in single automaton model. A firing of t_1, t_2, t_3 is represented as edge (A), (B), (C). A global clock `gc1` is assigned for the source transition t_1. Clocks `c[]` and variables `xf[]` are assigned for signals. k is the maximum number of signals in the pathway. `xf[`i`]` denotes the firing delay time of the i-th signal. Multiple signals are represented as a single location with `c[]` and `xf[]`. Firing delay time of the i-th signal is implemented by using location invariant `(c[`i`]<=xf[`i`])` and guard `(c[`i`]>=xf[`i`])`. In a firing of transition t_1 (Edge (A)), action `(xf[tf1]:=9)` means setting the firing delay time of next transition t_2. `tf`i denotes the signal ID of the first reserved signal in transition t_i. Action `(tx2[i2]:=tf1)` means reserving the signal that fires on t_1 to next transition t_2. `tx`i`[`k`]` stores signal IDs reserved in transition t_i. `(tf2:=tx2[0])` removes the first reserved signal ID of transition t_2. In a firing of the transition t_2 (Edge (B)), action `(mp1--, mp2++, c[tf2]:=0, xf[tf2]:=0, tx3[i3]:=tf2, tf3:=tx3[0])`

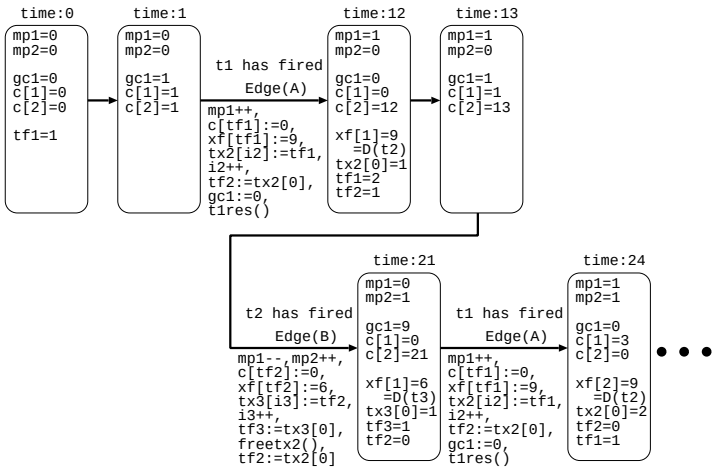


Fig. 5. The state transition of single automaton model of $TPNR_1$

is the same role as above. `freetx2()` organizes reserved signal IDs of t_2. `xf[0]` denotes the end time. In this model, we can analyze the model until `xf[0]`.

3.2 Transformation algorithms

In this subsection, we give, for each modeling method, an algorithm for transforming a Petri net to an automaton model. Due to limitations of space, we restrict the algorithm to the modeling of a single path Petri net model.

(1) Multiple automata modeling method
<<Transformation to Multiple Automata Model>>
Input: Petri net model $TPNR = (P, T, \mathcal{E}, D, R)$, number n of signals
Output: Multiple automata model A_1, A_2, $\cdots$, A_n

Foreach $i = 1$ to n, make A_i according to the following:

1. $L \leftarrow P \cup T$
2. $C \leftarrow \{$`gc1`, `c`$i\}$
3. $E \leftarrow \{(p, \emptyset,$(`c`$i$`>=` $D(t)), \emptyset, t) |\ (p,t) \in A\}$
   $\cup \{(t, ($`gc1:=0`, `c`i`:=0`$), ($`gc1>=`$D(t)), \{$`gc1`, `c`$i\}, p) |\ |^\bullet t| = 0, (t,p) \in A\}$
   $\cup \{(t, (c i := 0), \emptyset, \{$`c`$i\}, p) |\ |t^\bullet| > 0, (t,p) \in A\}$
4. $I \leftarrow \{(t, ($`gc1<=`$D(t_1))) |\ t \in T, |^\bullet t| = 0\}$
   $\cup \{(t, ($`c`i`<=`$D(t))) |\ t \in T, |t^\bullet| > 0\}$
   $\cup \{(p, ($`c`i`<=`$D(t))) |\ p \in P$, t is the output transition of $p\}$

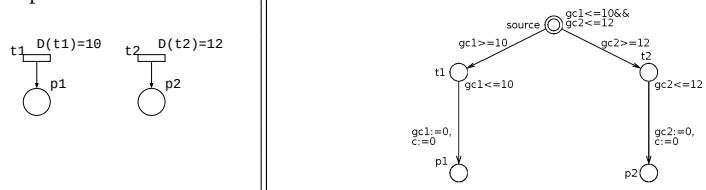
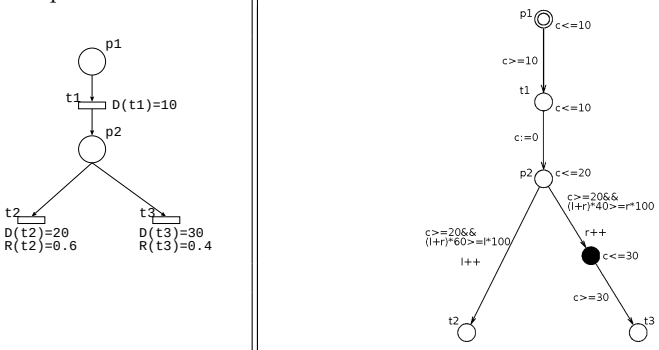
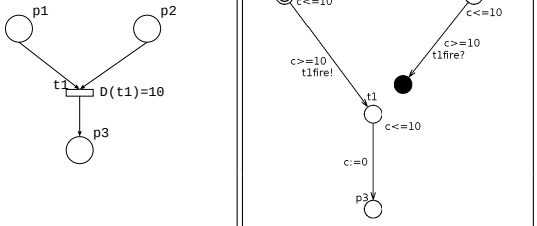
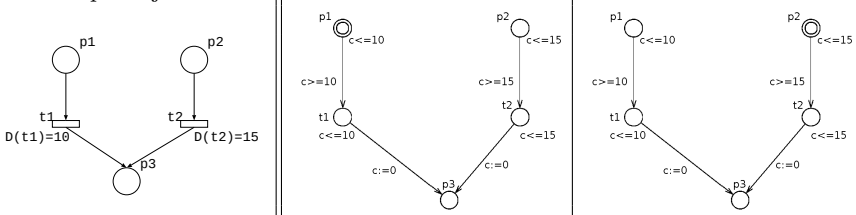
(2) Single automaton modeling method
<<Transformation to Single Automaton Model>>
Input: Petri net model $TPNR = (P, T, \mathcal{E}, D, R)$, number n of signals
Output: Single automaton model A, variables `mp1`, `mp2`, $\cdots$, `mp`$|P|$, `tf1`, `tf2`, $\cdots$, `tf|T|`, `xf[]`, `tx1[]`, `tx2[]`, $\cdots$, `tx`$|T|$`[]`.

1. $L \leftarrow \{l_0\}$
2. $C \leftarrow \{$`gc1`, `c[1]`, `c[2]`,$\cdots$, `c[k]`$\}$
3. $E \leftarrow \{(l_0, ($`mp1++`, `c[tf1]:=0`, `xf[tf1]:=`$D(t_2)$, `tx2[i2]:=tf1`, `i2++`, `tf2:=tx2[0]`, `gc1:=0`, `t1res()`$), ($`gc1>=`$D(t_1)), \{$`gc1`, `c[tf1]`$\}, l_0)\}$
   $\cup \{(l_0, ($`mp`$|P|$`--`, `c[tf|T|]:=0`, `xf[tf`$|T|$`]:=10000`, `freetx|T|()`, `tf`$|T|$`:=tx`$|T|$`[0]`$), ($`c[tf`$|T|$`]>=xf[tf`$|T|$`] && xf[tf`$|T|$`]==`$D(t_{|T|}))), \{$`c[tf`$|T|$`]`$\}, l_0)\}$
   $\cup \bigcup_{i=2to|T|-1} \{(l_0, ($`mp`$(i-1)$`--`, `mp`$i$`++`, `c[tf`$i$`]:=0`, `xf[tf`$i$`]:=`$D(t_{(i+1)})$,`tx`$(i+1)$`[i`$(i+1)$`]:=tf`$i$,`i`$(i+1)$`++`, `tf`$(i+1)$`:=tx`$(i+1)$`[0]`, `freetx`$i$`()`, `tf`$i$`:=tx`$i$`[0]`$), ($`c[tf`$i$`]>=xf[tf`$i$`]&&xf[tf`$i$`]==`$D(t_i)), \{$`c[tf`$i$`]`$\}, l_0)\}$
   `t1res()` resets `c[]` and `xf[]`.
   `freetx`i`()` organizes reserved signal IDs of transition t_i.
4. $I \leftarrow \{(l_0, ($`c[0]<=xf[0]&&c[1]<=xf[1]&&`$\cdots$ `&&c[k]<=xf[k]&& gc1<=`$D(t_1)))\}$

3.3 Pattern lists

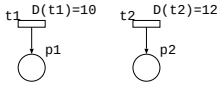
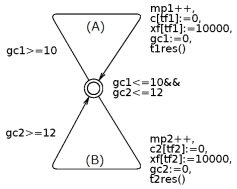
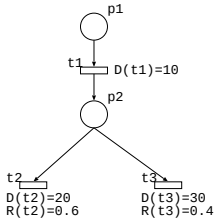
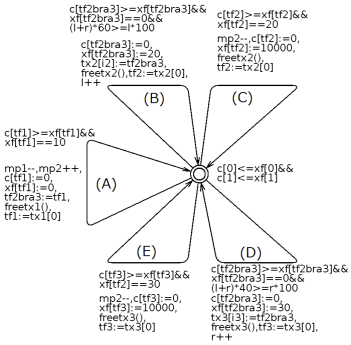
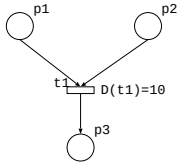
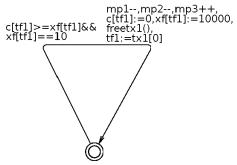
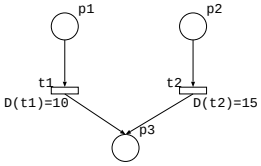
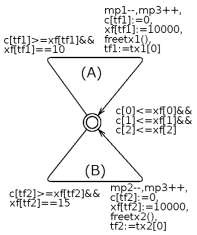
We give the pattern lists of transforming TPNR to multiple automata model and single automaton model to ease the transformation. Multiple automata model

Table 2. The pattern list of transforming TPNR to multiple automata model

TPNR model	multiple automata model
Multiple source transitions	
A place branch	
A transition join	
A place join	

and single automaton model can be created by combination of these patterns. Table 2 is the pattern list of transforming TPNR to multiple automata model. `c` is a local clock and `gc` is a global clock.

Table 3. The pattern list of transforming TPNR to single automaton model

TPNR model	single automaton model
Multiple source transitions 	
A place branch 	
A transition join 	
A place join 	

- The first column is for multiple source transitions. Location `source` is a global source. This location implements all of the firing delay times of source transition with global clocks, `gc1`, `gc2`.
- The second column is for a place branch. A place branch is implemented by using `l` and `r`. `l` denotes the firing count of `t2`, and `r` denotes the firing count of `t3`. And guard (`(l+r)*40>=l*100`) limits that firing count of `t2`. `t3` is implemented by a similar way of `t2` but firing delay time of `t3` is longer than that of `t2`. So, the blacken location is added to implement the firing delay time.
- The third column is for a transition join. A transition join is implemented by channels `t1fire!` and `t1fire?`. A signal on `p1` and a signal on `p2` are synchronized, then two signals fires on `t1`. The blacken location is added to abandon the signal on `p2`.
- The fourth column is for a place join. A place join is implemented by a similar way of <<Transformation to Multiple Automata Model>>.

Table 3 is the pattern list of transforming TPNR to single automaton model. `c` is a local clock and `gc` is a global clock.

- The first column is for multiple source transitions. Edge (A) implements a firing of t_1 and edge (B) implements a firing of t_2. Firing delay times of t_1 and t_2 are implemented by global clocks `gc1` and `gc2`.
- The second column is for a place branch. Edge (A) implements the firing of transition t_1 with substituting 0 in `xf[]` and `tf2bra3` stores the signal ID that isn't reserved. Edges (B) and (D) are limiting the firing count by similar way of multiple automata model and reserve a signal to transition t_2 or t_3. Edges (C) and (E) implement the firing of transitions t_2 and t_3.
- The third column is for a transition join. A transition join is implemented by only updating variables `mp1` and `mp2`.
- The fourth column is for a place join. A place join is implemented by a similar way of <<Transformation to Single Automaton Model>>. Edge (A) implements the firing of transition t_1 and edge (B) implements the firing of transition t_2.

4 Application example: IL-1 signaling pathway

In this section, we apply the proposed modeling methods to IL-1 signaling pathway. And we compare the obtained models and discuss the merits and demerits of the modeling methods.

IL-1 is a proinflammatory cytokine, and plays an important role in regulating the mechanism of proinflammatory. There cause multiple signals flowing in the signaling pathway because a ligand joins a receptor repeatedly. The whole Petri net model of IL-1 signaling pathway is shown in Fig.3 of [9].

The correctness of the model must be examined. We can check the correctness of the model by model checking. For example,

– We can check the reachability with time concept. Checking the reachability helps us to understand signaling pathways. Let c be a clock, we write a TCTL expression of this property as `E<>` $M(p_1)$`>0 &&` c`==30`. This expression means there exists a path where a signal is on p_1 when c is 30.
– We can also check that there is no retention in the model. Retention means that signals are accumulated at any place. We write a TCTL expression of this property as `A[]` $M(p_1)$`<=5`. This expression means that the number $M(p_1)$ of signals for place p_1 is always 5 or less.

In this paper, we focus on checking the retention. If there is retention in the Petri net model, we consider the error in the Petri net model or the possible of unknown paths in the signaling pathway.

We apply two modeling methods to a part of IL-1, which is shown in Fig.1. The size of the Petri net model is $|P| = 7$, $|T| = 12$. Figure 6 shows the multiple automata model of Fig.1. Figure 7 shows the single automaton model of Fig.1.

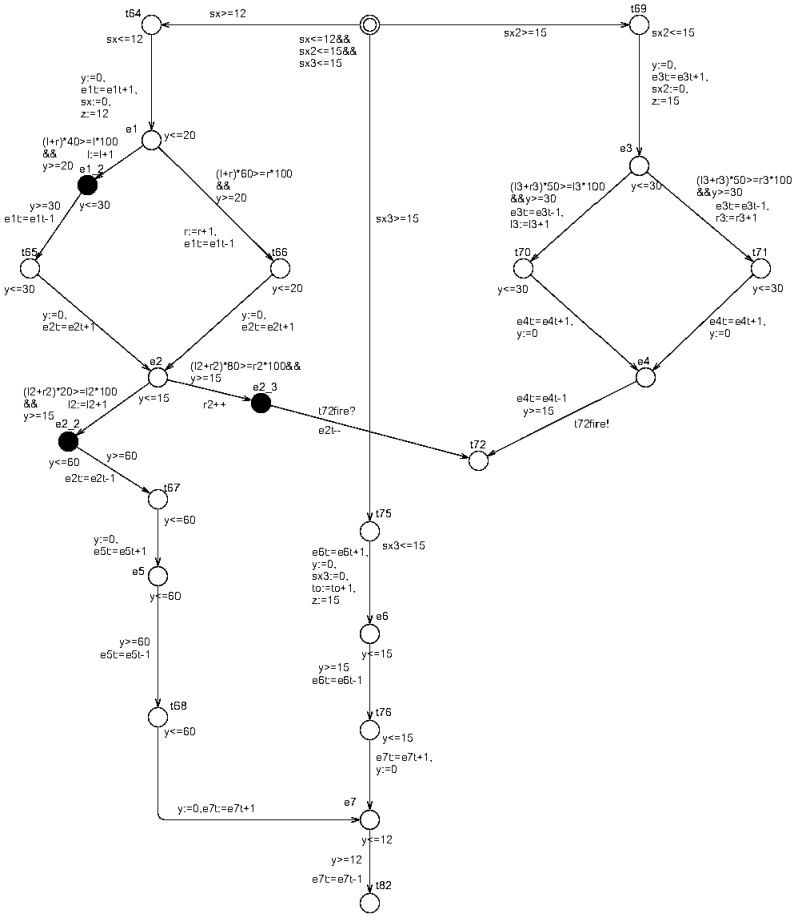


Fig. 6. The multiple automata model of the part of IL-1 signaling pathway. Each automaton represents the behavior of a single signal.

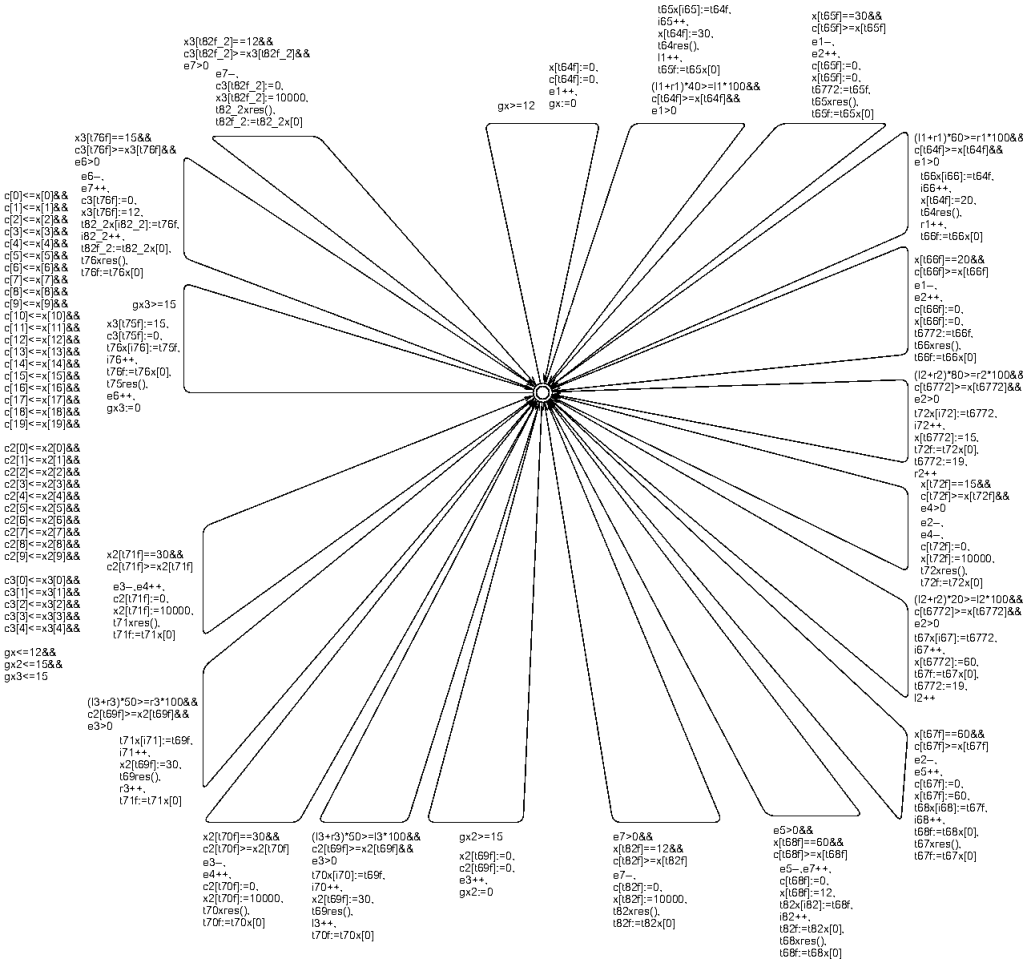


Fig. 7. The single automaton model of the part of IL-1 signaling pathway. This automaton represents the behavior of multiple signals.

This Petri net model includes three source transitions, therefore we applied the first column of each pattern list at the case of three source transitions. Places e_1, e_2, and e_3 have two output transitions, therefore we applied the second column of each pattern list. We applied the third column of each pattern list for transition t_{72} because t_{72} has two input places. And we applied the fourth column of each pattern list for place e_7 because e_7 has two input places.

Table 4 shows the size and the number of clocks and variables of the obtained models. The size of each automaton of the multiple automata model is $|L| = 23$, and $|E| = 26$. In comparison, the size of the single automaton model is $|L| = 1$, and $|E| = 19$. The number of clocks of the multiple automata model is $3+n$, and the number of variables is 6, where n is the number of signals. The number of clocks of the single automaton model is 48, and the number of variables is

Table 4. Evaluation results of example 1: IL-1 signaling pathway $|P|$=7, $|T|$=12

Model	Number of signals n	Number of clocks	Number of states explored	Checking time(min)
Multiple	39	42	139858	2
automata	78	81	770854	110
$\|L\|$=23, $\|E\|$=26	128	131	2205987	180
variables:6	129	132	Out of memory	
Single	107	48	581918	5
automaton	325	48	2393918	20
$\|L\|$=1, $\|E\|$=19	702	48	5505758	35
variables:53	703	48	Out of memory	

40. In addition, the number of arrays is 13. The number of clocks continue to increase according to the number of signals n in the multi automata model, but the number of clocks is constant in the single automaton model. The size of the single automaton model is smaller than the size of the multiple automata model, but the number of variables of single automaton model is larger than that of the multiple automata model under the same expressive power.

We can analyze the retention property by using those automaton models. Table 4 shows the number of signals, the number of states explored, checking time. Model checking is performed on the PC with CPU Xeon 2.13GHz and memory 3.2Gbyte. We could check the retention property of the multiple automata model until the number of signals is 128. Meanwhile we could check that of the single automaton model until the number of signals is 702. The single automaton model enables us to analyze more signals than the multiple automata model. The number of states explored of the single automaton model is smaller than the multiple automata model, and checking time is also shorter.

5 Conclusion

In this paper, we proposed two modeling methods for signaling pathways with multiple signals. Those modeling methods use different representation model. Next we gave for each representation model, a transformation algorithm, and a pattern list. Then we applied these proposed methods to the IL-1 signaling pathway. The results show that signaling pathway with multiple signals should be represented as not automata but variables, and the model size to be analyzed can be increased by devising of modeling method. We have also applied these method to endocytosis signaling pathway, and a similar trend have been obtained.

As future work, we plan to develop a method treating transition branches. An approach is to prepare child process for implementing parallel part and devising the clock handling. And we think that it is easy to extend our method to time Petri net model. We will verify that our method can be applied to time Petri net model.

Acknowledgement
The authors would like to thank Prof. Hiroshi Matsuno, Prof. Qi-Wei Ge, and Mr. Yuki Murakami for their valuable advices.

References

1. N.Chabrier, F.Fages, "Symbolic model checking of biochemical networks," Lecture Notes in Computer Science 2602, pp.149–162, 2003.
2. W. J. Bos , "Interactive Signaling Network Analysis Tool," Master Thesis, University of Twente, 2009.
3. M. Kwiatkowska, G. Norman, D. Parker, "Probabilistic model checking for systems biology," Symbolic Systems Biology, Jones and Bartlett, 2010.
4. F. Cassez, O. H. Roux, "From Time Petri Nets to Timed Automata", Petri Net, Theory and Applications, InTech Education and Publishing, 2008.
5. C. Li, S. Suzuki, Q. W. Ge, M. Nakata, H. Matsuno, S. Miyano, "Structural modeling and analysis of signaling pathways based on Petri nets," Journal of Bioinformatics and Computational Biology, vol.4, no.5, pp.1119–1140, 2006.
6. H. Matsuno, C. Li, S. Miyano, "Petri net based descriptions for systematic understanding of biological pathways," IEICE Trans. Fundamentals, vol. E89-A, no. 11, pp. 3166–3174, 2006.
7. UPPAAL Group, UPPAAL, http://www. uppaal. com/, 1995.
8. UPPAAL Group, A Tutorial on Uppaal, http://www.cs.aau.dk/∼adavid/ RTSS05/ UPPAAL-tutorial.pdf, 2005.
9. Y. Miwa, Y. Murakami, Q. W. Ge, C. Li, H. Matsuno, S. Miyano, "Delay Time Determination for the Timed Petri Net Model of a Signaling Pathway Based on Its Structural Information," IEICE Trans. Fundamentals, vol.E93-A, no.12, pp.2717–2729, 2010.
10. T. Murata, "Petri nets: Properties, analysis and applications," Proc. of the IEEE, vol.77, no.4, pp.541–580, 1989.
11. V. N. Reddy, M. L. Mavrovouniotis, M. N. Liebman, "Petri net representations in metabolic pathways," Proc. Int. Conf. Intell. Syst. Mol. Biol., vol.1, pp. 328–336, 1993.
12. M. Calder, V. Vyshemirsky, D. Gilbert and R. Orton, "Analysis of signalling pathways using the PRISM model checker," Proc. Computational Methods in Systems Biology (CMSB'05), pp. 179–190, 2005.
13. M. Heiner, I. Koch and J. Will, "Model Validation of Biological Pathways Using Petri Nets–demonstrated for apoptosis," Biosystems, vol.75, no.1–3, pp.15–28, 2004.
14. D. Gilbert and M. Heiner, "From Petri nets to differential equations – anintegrative approach for biochemical network analysis," Proc. 27th International Conference on Application and Theory of Petri Nets, LNCS 4024, pp.181–200, 2006.
Biosystems, vol.75, no.1–3, pp.15–28, 2004.

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.102-116

MPath2PN - Translating metabolic pathways into Petri nets

Paolo Baldan[1], Nicoletta Cocco[2], Francesco De Nes[2], Mercè Llabrés Segura[3], Andrea Marin[2], Marta Simeoni[2]

[1]Università di Padova, Italy
baldan@math.unipd.it
[2]Università Ca' Foscari di Venezia, Italy
{cocco,marin,simeoni}@dais.unive.it, kekodenes@gmail.com
[3]Universitat de les Illes Balears, Spain
merce.llabres@uib.es

Abstract. We propose *MPath2PN*, a tool which automatically translates metabolic pathways, as described in the major biological databases, into corresponding Petri net representations. The aim is to allow for a systematic reuse, in the setting of metabolic pathways, of the variety of tools existing for Petri net analysis and simulation. The current prototype implementation of *MPath2PN* inputs the KEGG description of a metabolic pathway and produces two Petri nets, mainly differing for the treatment of ubiquitous substances. Such Petri nets are represented using PNML, a standard format for many Petri net tools. We are extending the tool by considering further formats for metabolic pathways in input and for Petri nets in output. *MPath2PN* is part of a more general project aimed at developing an integrated framework which should offer the possibility of automatically querying databases for metabolic pathways, producing corresponding Petri net models and performing analysis and simulation on them by means of various tools.

1 Introduction

Metabolic pathways are complex systems whose understanding is important in many fields, in particular in biology and medicine. Various techniques have been proposed to model and analyse metabolic pathways. Among these, Petri nets are a well-known formalism, used in computer science for modelling concurrent and distributed systems, which turns out to be particularly natural for representing metabolic pathways and with the advantage of the availability of many tools for visualisation, simulation and analysis. By using Petri nets it is possible to represent and analyse fundamental properties of metabolic pathways, like conservation relations on metabolites (corresponding to P-invariants), steady state flux distributions (corresponding to T-invariants), the rates of chemical reactions (corresponding to marking dependent rates in continuous transitions) or control mechanisms, such as positive or negative feedbacks.

When modelling a metabolic pathway as a Petri net one has to face several problems related to the multiplicity of data sources and formats. On the one

hand, the information on the pathway may be stored in different databases each using its own data format. On the other hand, once constructed, the Petri net model could be analysed with different Petri net tools, each one having its specific input format. Our proposal is aimed at alleviating this problem, automatising the recovery of metabolic data and their translation into corresponding Petri net models, which can be encoded using the input format of different tools available for Petri nets. This is part of a larger project - in progress - aimed at developing a framework able to automatically retrieve metabolic data from the web, produce corresponding Petri net representations and analyse them through the available tools. The framework should deal with the various databases for metabolic pathways and the different tools for Petri nets.

In this paper we present a prototype implementation of the automatic translation of the metabolic data into a Petri net model. The tool, *MPath2PN*, is written in Java and it is conceived to deal with different translations, that is different databases in input, such as KEGG and the BioModels Database, and different Petri net tools in output. At present it includes two specific translations from KEGG's data to PNML for PIPE2. The first translation is rather efficient since it considers a KGML file as the main source. The second translation is slower, since it gets most of the input data from the KEGG web service, but it provides a more detailed representation of the pathway which includes also ubiquitous substances.

The paper is organised as follows. In Section 2 we give a brief introduction to metabolic pathways and their main databases. In Section 3 we recall how to give a Petri net representation of a metabolic pathway. In Section 4 we describe the tool structure and the two translations from KEGG to PNML for PIPE2. Finally, in Section 5 we draw some conclusions.

2 Metabolic Pathways

An organism depends on its metabolism, the chemical system which generates the essential components for life and the energy necessary to synthesise and use them. Subsystems dealing with some specific function are called *metabolic pathways*. Biologists usually represent a metabolic pathway as a network of *chemical reactions*, catalysed by one or more *enzymes*, where some molecules (*reactants* or *substrate*) are transformed into others (*products*). Enzymes are not consumed in a reaction, even if they are necessary and used while the reaction takes place. The product of a reaction is the substrate of the next one.

To characterise a metabolic pathway, it is necessary to identify its components (namely the reactions, enzymes, reactants and products) and their relations. Such relations can be represented through a *stoichiometric matrix*. An element of the matrix, a stoichiometric coefficient n_{ij}, represents the degree to which the i-th chemical species participates in the j-th reaction. The kinetic of a pathway is determined by the *rate* associated with each reaction. It is represented by a rate equation, which depends on the concentrations of the reactants and on a reaction

rate coefficient (or rate constant) which includes all the other parameters (except for concentrations) affecting the rate.

A metabolic pathway contains many steps, one is usually irreversible, the other steps are usually reversible and in many cases the pathway can go in the opposite direction depending on the needs of the organism. *Glycolysis* is a good example of this behaviour: it is a fundamental pathway which converts glucose into pyruvate and releases energy. When glucose enters a cell, it is phosphorylated by ATP to glucose 6-phosphate in a first irreversible step, thus glucose will not leave the cell. When there is an excess of energy, the reverse process, the *gluconeogenesis*, converts pyruvate into glucose: glucose 6-phosphate is produced and stored as glycogen or starch. Most steps in gluconeogenesis are the reverse of those found in glycolysis, but the three reactions of glycolysis producing most energy are replaced with more kinetically favorable reactions. This system allows glycolysis and gluconeogenesis to inhibit each other.

Information on metabolic pathways are collected in many different databases. The **KEGG PATHWAY** database [9] contains the main known metabolic, regulatory and genetic pathways for different species. It integrates genomic, chemical and systemic functional information [38]. KEGG can be queried through a language based on XML [6], called **KGML** (KEGG Markup Language) [8], but also a web service for querying the system from users programs is available. Another important repository is the **BioModels Database** in the SBML.org site [17]. The models are coded in **SBML** (Systems Biology Markup Language), a language based on XML. Other free access databases are **MetaCyc** [11, 24], **Reactome** [15], **TRANSPATH**, which is part of **BIOBASE** [20] and **BioCarta** [1]. Relevant information can be found also in other databases, such as **BRENDA** [3, 25], **ENZYME** [5], **DIP** [4, 52], **MINT** [12, 27] and **BIND**.

3 Petri nets for modelling Metabolic Pathways

In some seminal papers Reddy et al. [50, 48, 49] and Hofestädt [36] propose Petri nets (PNs) for representing and analysing metabolic pathways. Since then a wide range of literature has grown on the topic (see, e. g., [26, 39, 21] for surveys on modelling metabolic pathways through PNs). PNs are a well-known formalism applied in computer science for modelling concurrent systems. They have an intuitive graphical representation which may help the understanding of the modelled system, a sound theory and many applications both in computer science and in real life systems (see [45, 51, 44, 28] for surveys on PNs and their properties). A PN model can be decomposed in order to master the overall complexity and it enables a large number of different analyses. Just to mention a few, one can determine conflicting evolutions, reachable states, cycles, states of equilibrium, bottlenecks or accumulation points. Additionally, once a qualitative PN model has been devised, quantitative information can be added incrementally. PNs seem to be particularly natural for representing metabolic pathways, as there are many similarities between concepts in biochemical networks and in PNs. They both consist of collections of reactions which consume and pro-

duce resources and their graphical representations are similar. This suggest to exploit the techniques developed for PNs also for metabolic pathways. In fact many tools are available for visualisation, analysis and simulation of PNs, a quite comprehensive list can be found at the Petri net World site [14].

Several generalisations of the basic PN formalism have been proposed to better modelling biological systems (such as PNs with test and inhibitor arcs [42, 43], Coloured PNs [35, 54], Timed PNs [29, 34, 46], Stochastic PNs [32, 41, 33], Continuous PNs [30, 23, 33, 39] and Hybrid PNs [42, 43]). Some extensions concern the qualitative aspects of the models and aim at increasing the expressive power or the modelling capabilities of the formalism. Other extensions introduce quantitative concepts, such as time and probability, thus allowing for the representation of temporal and stochastic aspects of biological systems, respectively. In this paper we will be concerned only with basic PNs, used for a qualitative modelling of metabolic pathways.

3.1 Petri net representation of a metabolic pathway

The qualitative representation of a metabolic pathway by means of a PN can be derived by exploiting the natural correspondence between PNs and biochemical networks. In fact, places in PNs are associated with molecular species, such as metabolites, proteins or enzymes; transitions in PNs correspond to chemical reactions; input places represent the substrate or reactants; output places represent reaction products. The incidence matrix of the PN is identical to the stoichiometric matrix of the system of chemical reactions. The number of tokens in each place of the PN indicates the amount of substance associated with that place. It may represent either the number of molecules expressed in moles or the level of concentration, suitably discretised by introducing a concept of concentration level [31].

Although the correspondence between metabolic pathways and PN elements is rather straightforward, some modelling choices have to be taken in the construction of a PN representation of a metabolic pathway. For example, enzymes and ubiquitous substances, such that H_2O, phosphate, ADP and ATP, might not be represented in the PN. Enzymes are taken and then released by the reactions and they are usually not represented in the PN model. This is an appropriate choice as long as their concentration do not change. Also ubiquitous substances, once assumed to be constant, can be omitted in the PN model. In this way the resulting model is greatly simplified, but, as an obvious drawback, processes involving such substances, such as the energy balance, are not modelled. In the PN models produced by the current prototype enzymes are not explicitly represented. Instead, as clarified later, the decision on whether to include information on the ubiquitous substances is left to the user.

Additionally, in a metabolic pathway one can distinguish between internal and external metabolites. The former are entirely produced and consumed in the network, while the latter represent sources or sinks, that is, connection points with other pathways producing or consuming them. External metabolites can be represented in the PN model in different ways, with different impacts on the

resulting net. In the translations currently performed by the prototype, external metabolites will simply result in places where connected transitions either all consume or all produce tokens. Their special status may be considered later in the simulation or analysis phase.

Another modelling problem arises from the fact that most of the reactions in a pathway are reversible. A reversible reaction is decomposed into two distinct reactions, a forward one and a backward one, leading to two corresponding transitions in the PN model. If the PN model does not represent the kinetic factors, the presence of the forward and backward transitions leads to a cyclic behaviour producing and destroying the same molecules, which might not be of biological interest. In the current implementation pairs of transitions corresponding to reversible reactions can be distinguished by their identifiers, so that the corresponding cyclic behaviours may be filtered out, if desired, in the analysis or simulation phase (e.g., an analysis based on T-invariants could ignore the trivial invariants consisting of pairs of transitions generated by a reversible reaction).

Once we have a qualitative model, quantitative data can be added to refine the representation of the behaviour of the pathway. In particular, extended PNs may have an associated transition rate which depends on the kinetic law of the corresponding reaction. This introduces further representation problems and choices, but in this paper we consider only qualitative modelling. A more detailed description of the representation of metabolic pathways with PNs can be found in [39, 21], where qualitative and quantitative modelling aspects are discussed and analysed.

4 The tool *MPath2PN*

The tool *MPath2PN* is intended to provide a way of automatically transforming a metabolic pathway, expressed in one of the various existing formalisms (e.g. KGML, SBML), into a corresponding PN, also expressed in one of the existing formalisms (e.g. PNML [13], a standard format used by many analysis tools for PNs, or the specific input formalism for PN tools, such as SNOOPY [18], INA [53] or TimeNET [19]).

We developed a prototype in Java with a structure which is modular enough to cope with many different translations (see Figure 1). We also implemented two specific translations which follow the modelling choices described in Section 3.1. Both of them derive the description of a metabolic pathway from the KEGG database and generate a corresponding PN. A basic source of information on the pathway is a file, in KGML format, which can be downloaded from KEGG. A file describing the corresponding PN model is produced, in PNML format for PIPE2 (Platform Independent Petri net Editor 2) [22], an open source platform independent tool for creating and analysing PNs. The two translations differ for the level of detail of the description of the pathway: the second translation considers also the presence of ubiquitous substances.

Since most of the descriptions of metabolic pathways and of PNs are based on XML formats, *MPath2PN* produces the translation by using XSLT (eXtensible

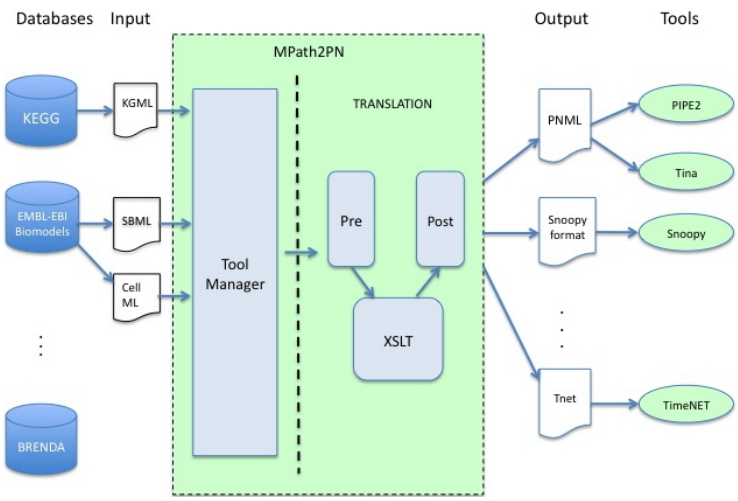


Fig. 1. Structure of *MPath2PN*

Stylesheet Language Transformation [7]) in the Saxon [16] open source version. Each translation requires the definition of an appropriate style sheet XSL which specifies the translation rules to be applied. Often there is the need to integrate various information in the translation, hence a translation is standardised into a three step process: pre-treatment, XSL translation and post-treatment. For the pre- and post-treatment, Java classes can be developed which modify respectively the input and the output files.

4.1 The first translation from KGML to PNML for PIPE2

The first translation implemented in *MPath2PN* consists of a plain transformation from a source KGML file describing the pathway downloaded from KEGG, to a target file describing the produced PN in PNML format for PIPE2.

Consider for example the KEGG pathway of the *Glycolysis / Gluconeogenesis* in *Homo sapiens* shown in Figure 2. We enclosed in a shaded box a small part of the pathway corresponding to a single reversible reaction, i.e., *β-D-glucose 6-phosphate ketol isomerase* (R03321). The KEGG page relative to such reaction is shown in Figure 3. The reaction is catalysed by the enzyme identified by the EC number 5.3.1.9 and it involves the compounds *β-D-glucose 6-phosphate* (C01172) and *β-D-Fructose 6-phosphate* (C05345). Note that KEGG uses its own identifiers for reactions and compounds. Let us take reaction R03321 as a running example for the translation from KGML to PNML.

The structure of the KGML format is shown in Figure 4. The root node represents the complete pathway, which is composed by nodes **entry**, **relation** and **reaction**, all with multiplicity $0, .., \infty$. A node **entry** represents a node in the KEGG pathway such as a compound, an enzyme or also a reference to another pathway. A node **relation** represents a relation between two proteins,

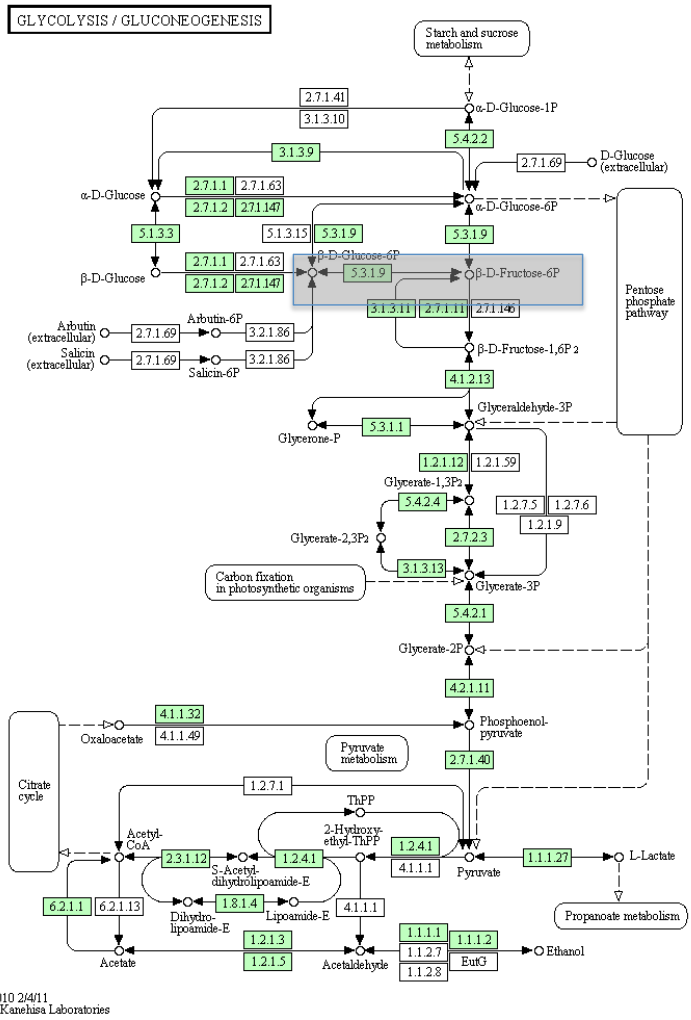


Fig. 2. KEGG pathway of the *Glycolysis / Gluconeogenesis* in *Homo sapiens*

or between a protein and a compound, or also a link to another map. A node **reaction** represents a pathway's reaction without dynamic information.

For instance, compound C01172 and reaction R03321 of our running example are represented in KGML as follows:

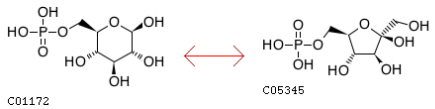
Entry	R03321 Reaction
Name	beta-D-Glucose 6-phosphate ketol-isomerase
Definition	beta-D-Glucose 6-phosphate <=> beta-D-Fructose 6-phosphate
Equation	C01172 <=> C05345
	 C01172 C05345
RPair	RP02940 C01172_C05345 main
Enzyme	5.3.1.9
Pathway	rn00010 Glycolysis / Gluconeogenesis rn01100 Metabolic pathways rn01110 Biosynthesis of secondary metabolites rn01120 Microbial metabolism in diverse environments
Orthology	K01810 glucose-6-phosphate isomerase [EC:5.3.1.9] K06859 glucose-6-phosphate isomerase, archaeal [EC:5.3.1.9] K13810 transaldolase / glucose-6-phosphate isomerase [EC:2.2.1.2 5.3.1.9]

Fig. 3. The KEGG page of reaction R03321

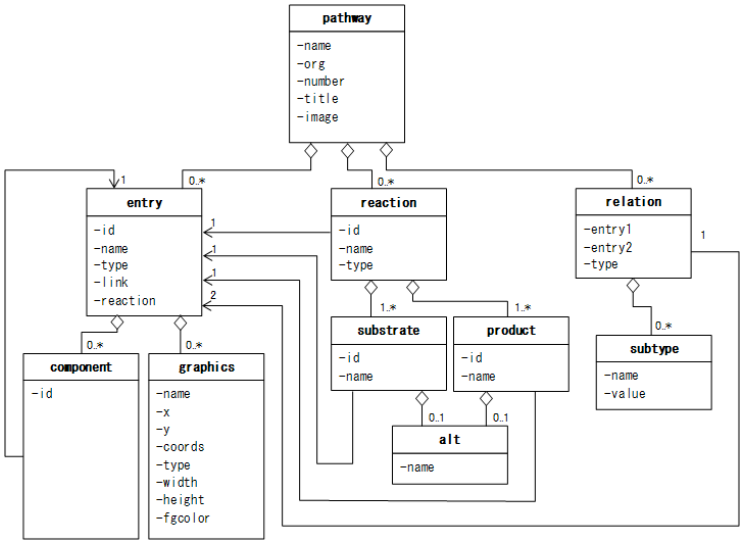


Fig. 4. Structure of a KGML file.

```
<entry id="90" name="cpd:C01172" type="compound"
    link="http://www.kegg.jp/dbget-bin/www_bget?C01172">
    <graphics name="C01172" x="332" y="301" type="circle" width="8"
    height="8" fgcolor="#000000" bgcolor="#FFFFFF"/>
</entry>
<reaction name="rn:R03321" type="reversible">
    <substrate name="cpd:C01172"/>
    <product name="cpd:C05345"/>
</reaction>
```

To build the PN representation of the pathway we use the nodes **entry** and **reaction**. Compounds correspond to places in the PN and reactions to transitions. The arcs are obtained by inspecting substrates and products in reactions.

The style sheet **net.xsl** implements most of the translation. It uses other XSLs dealing with the various components: **labels.xsl**, **places.xsl**, **transitions.xsl** and **arcs.xsl**, as shown in Figure 5.

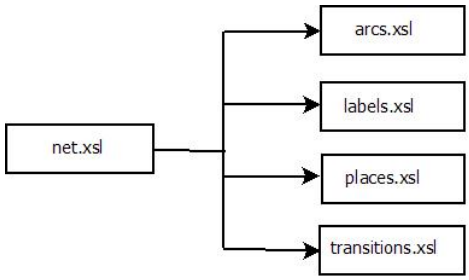


Fig. 5. The PNML style sheet structure

In **places.xsl** the entries are checked to determine whether they have to be translated into places: only compounds are represented. The target code generated for compound C01172 of our running example is the following:

```
<place id="cpd:C01172">
    <graphics><position x="332" y="301"/></graphics>
    <name><value>cpd:C01172</value></name>
</place>
```

Transitions are created by **transitions.xsl** from the **reaction** nodes in the KGML format. As already mentioned, a non-reversible reaction produces a single transition, while a reversible reaction produces two transitions (a direct and an inverse one). The inverse transition is identified by the fact that its *id* obtained from the *id* of the direct transition by adding the string "♯**rev**" as suffix. This allows to recognise cycles in the behaviour introduced by this encoding of reversible reactions.

The PNML code generated for reaction R03321 is the following:

```
<transition id="rn:R03321">
    <name><value>rn:R03321</value></name>
    <rate><value>1.0</value></rate>
    <timed><value>false</value></timed>
</transition>
<transition id="rn:R03321#rev">
    <name><value>rn:R03321#rev</value></name>
    <rate><value>1.0</value></rate>
    <timed><value>false</value></timed>
</transition>
```

The arcs are generated by templates in **arcs.xsl**. They are inferred by the nodes **reaction** and their children **substrate** and **product** in the KGML format. For each pair (substrate, product) the following arcs are created,

$$\text{substrate} \to \text{reaction}, \quad \text{reaction} \to \text{product},$$

and, obviously, if the reaction is reversible, we will have also the inverse arcs:

$$\text{inverse reaction} \to \text{substrate}, \quad \text{product} \to \text{inverse reaction}.$$

In our example the following arcs are generated in the target code:

```
<arc target="rn:R03321" source="cpd:C01172" id="cpd:C01172 to rn:R03321">
    <inscription><value>1</value></inscription>
    <type value="normal"/>
</arc>
<arc target="cpd:C05345" source="rn:R03321" id="rn:R03321 to cpd:C05345">
    <inscription><value>1</value></inscription>
    <type value="normal"/>
</arc>
<arc target="rn:R03321#rev" source="cpd:C05345" id="cpd:C05345 to rn:R03321#rev">
    <inscription><value>1</value></inscription>
    <type value="normal"/>
</arc>
<arc target="cpd:C01172" source="rn:R03321#rev" id="rn:R03321#rev to cpd:C01172">
    <inscription><value>1</value></inscription>
    <type value="normal"/>
</arc>
```

A KGML file representing a metabolic pathway does not provide any information on kinetic laws, initial concentrations of compounds and stoichiometric values. However, stoichiometric values, which are essential also for a qualitative modelling (they correspond to arc weights in the PN) can be retrieved through the KEGG web service. This is done in the post-treatment phase of the translation which, as a consequence of the multiple service invocations, is rather slow. In order to speed up this process, a caching of the information is introduced, so that each reaction is queried only once through the web service. Since KGML files do not provide information on ubiquitous substances, the resulting PN does not represent ubiquitous substances either.

The complete PN corresponding to the Glycolysis pathway of Figure 2, as it is visualised by PIPE2, can be found in Figure 6. The part corresponding to the running example is enclosed in the shaded box.

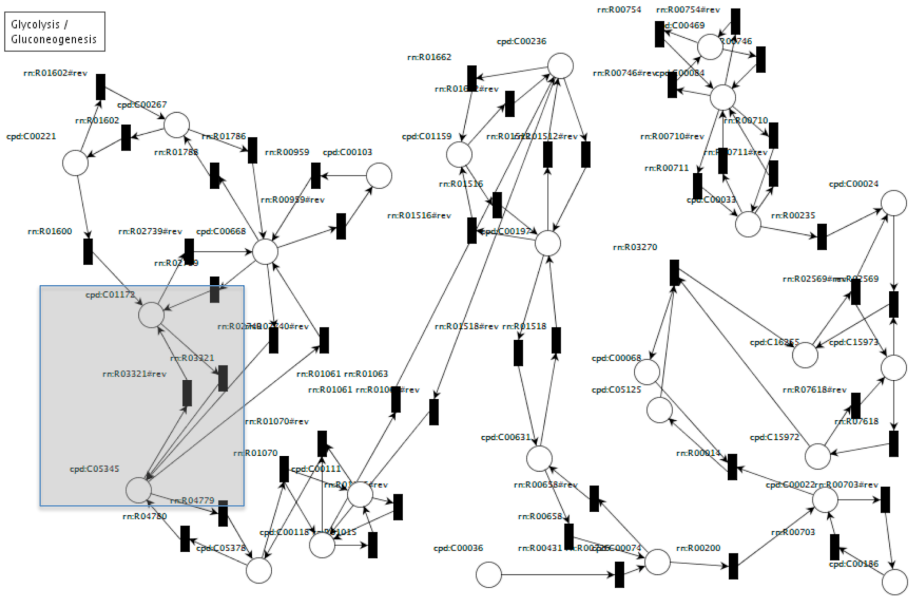


Fig. 6. Petri net resulting from the first translation of the *Glycolysis / Gluconeogenesis* in *Homo sapiens* (represented with PIPE2)

4.2 The second translation from KGML to PNML for PIPE2

The second translation also uses the basic KGML file describing the pathway, downloaded from KEGG but, in addition, it gets most of the input data from the KEGG web service. It is then much slower with respect to the first translation, but also more versatile since the data which can be accessed in this way are much more detailed. The pre-treatment phase is fundamental: it gets all the compounds from the stoichiometric formula of each reaction accessed through the web service. This permits also the representation of the ubiquitous compounds which are not present in the KGML file. Note that the KGML file is still necessary since it specifies, for example, if a reaction is reversible or not. Hence the data derived from the web service are inserted into the skeleton of the KGML file, which is then translated by means of the XSL style sheets defined in the first translation. The post-treatment phase is the same as in the first translation, but it is obviously faster, since stoichiometric formulas have been already cached and there is no need to access the web service for them.

5 Conclusions and future work

An obstacle to the use of PNs for modelling metabolic pathways seems to be, paradoxically, the amount of different sources of data on metabolic pathways and the number of simulation and analysis tools for PNs. This is due to the dishomogeneity both of databases formats for metabolic data and of input formats for PNs tools. To cope with this problem in the literature we find proposals for

- a standard format for metabolic data, such as SBML [17] or BioPAX [2], and a standard format for PN tools, such as PNML [13];
- unification or integration of different databases such as in [47] or [37], and translations between different data formats, such as in KEGGtranslator [10] or KGML2SBML and KGML2BioPAX [40].

In this paper we proposed a tool *MPath2PN*, for translating metabolic pathways into corresponding PN representations, coping with different input and output formats. The aim is to allow for a systematic reuse of the tools already developed for PNs also for the analysis and simulation of metabolic pathways. The input and output formats are generally based on XML. For this reason *MPath2PN* is based on XSLT and each translation can be defined by giving a corresponding style sheet XSL. Moreover *MPath2PN* allows for a pre-treatment and a post-treatment phase, implemented by Java classes, to permit the integration of different data sources on metabolic pathways.

We developed a prototype version of *MPath2PN* providing two rather standard translations. The first translation is from KGML to PNML for PIPE2 and it is rather efficient. The second translation is from KEGG to PNML for PIPE2 and it is slower, but it gives a more detailed representation of the pathway by considering also ubiquitous substances.

We are working on further translations to be included in MPath2PN:

- from KGML to the format of INA [53], a tool which allows for many different analysis of mainly qualitative Petri net models;
- from SBML to PNML;
- from SBML to the format of Snoopy [18], a tool which allows for analysis and simulation of stochastic/continuous PNs;
- from SBML to TimeNET/eDSPN format and from SBML to TimeNET/ SCPN format. TimeNET is a tool that allows for analysis and simulation of extended deterministic and stochastic Petri nets (eDSPN) and stochastic coloured Petri nets (SCPN). Using this tool, it is possible to specify transition rates that may depend on the global state of the net. As a consequence, the translation of the dynamic information from the SBML specification into the TimeNET format can be done efficiently and without the need of further assumptions, in a purely syntactical way.

Further extensions of *MPath2PN* consist in providing different translations between the same input and output formats in order to implement different modelling choices, for example we could represent explicitly also enzymes or supply different ways of dealing with external metabolites.

When quantitative data are available, as in SBML, it is possible to obtain a quantitative PN model of a metabolic pathway. In this case further modelling decisions have to be taken in the translation, such as whether to consider all modifiers (such as inhibitors and cofactors) or not, whether and how to scale or discretise the amounts of substances, which kinetic model to choose, and, more generally, whether to give a continuous, a discrete or a stochastic representation.

Mpath2PN is freely available at:
http://www.dsi.unive.it/∼simeoni/MPath2PNtool.tgz.

References

1. BioCarta: Charting Pathways of Life. http://www.biocarta.com.
2. BioPAX: Biological Pathway Exchange. http://www.biopax.org/index.php.
3. BRENDA: The Comprehensive Enzyme Information System. http://www.brenda-enzymes.info.
4. Database of Interacting Proteins. http://dip.doe-mbi.ucla.edu.
5. ENZYME: enzyme nomenclature database. http://www.expasy.ch/enzyme.
6. Extensible Markup Language. http://www.w3.org/XML.
7. Extensible Stylesheet Language Transformations. http://www.w3.org/TR/xslt.
8. Kegg Markup Language manual. http://www.genome.ad.jp/kegg/docs/xml.
9. KEGG pathway database. http://www.genome.jp/kegg/pathway.html.
10. KEGGtranslator. http://www.ra.cs.uni-tuebingen.de/software/KEGGtranslator/.
11. MetaCyc Encyclopedia of Metabolic Pathways. http://metacyc.org.
12. MINT: The Molecular INTeraction database. http://mint.bio.uniroma2.it.
13. Petri Net Markup Language. http://www.pnml.org.
14. Petri net tools. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools.
15. REACTOME. http://www.reactome.org.
16. SAXON: the XSLT and XQuery processor. http://saxon.sourceforge.net/.
17. SBML: Systems Biology Markup Language. http://sbml.org.
18. SNOOPY. http://www-dssz.informatik.tu-cottbus.de/index.html?/software/snoopy.html.
19. TimeNET. http://www.tu-ilmenau.de/fakia/TimeNet.timenet.0.html?&L=1.
20. TRANSPATH: The Pathway Database. http://www.biobase-international.com.
21. P. Baldan, N. Cocco, A. Marin, and M. Simeoni. Petri nets for modelling metabolic pathways: a survey. *Natural Computing*, 9(4):955–989, 2010. ISSN: 1567-7818.
22. P. Bonet, C.M. Llado, R. Puijaner, and W.J. Knottenbelt. PIPE v2.5: A Petri net tool for performance modelling. In *Proc. 23rd Latin American Conference on Informatics (CLEI 2007), San Jose, Costa Rica*. ACM, 2007.
23. R. Breitling, D. Gilbert, M. Heiner, and R. Orton. A structured approach for the engineering of biochemical network models, illustrated for signalling pathways. *Briefings in Bioinformatics*, 9(5):404–421, 2008.
24. R. Caspi, H. Foerster, C.A. Fulcher, P. Kaipa, M. Krummenacker, M. Latendresse, S. Paley, S. Y. Rhee, A. G. Shearer, C. Tissier, T. C. Walk, P. Zhang, and P. D. Karp. The MetaCyc Database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research*, 36(Database issue: D623-D631), 2008.
25. A. Chang, Scheer M., Grote A., I. Schomburg, and D. Schomburg. BRENDA, AMENDA and FRENDA the enzyme information system: new content and tools in 2009. *Nucleic Acids Research*, 37(Database issue: D588-D592), 2009.

26. C. Chaouiya. Petri net modelling of biological networks. *Briefings in Bioinformatics*, 8(4):210–219, 2007.
27. A. Chatraryamontri, A. Ceol, L. Montecchi Palazzi, G. Nardelli, M. V. Schneider, L. Castagnoli, and G. Cesareni. MINT: the Molecular INTeraction database. *Nucleic Acids Research*, 35(Database issue: D572-D574), 2007.
28. J. Esparza and M. Nielsen. Decidability issues for Petri Nets - a survey. *Journal Inform. Process. Cybernet. EIK*, 30(3):143–160, 1994.
29. H. Genrich, R. Küeffner, and K. Voss. Executable Petri Net Models for the Analysis of Metabolic Pathways. *Proceedings of the Workshop on Practical Use of High-level Petri Nets*, pages 1–14, 2000.
30. D. Gilbert and M. Heiner. From Petri Nets to Differential Equations - An Integrative Approach for Biochemical Networks Analysis. In *Petri Nets and Other Models of Concurrency - ICATPN 2006*, volume 4024 of *LNCS*, pages 181–200. Springer, 2006.
31. D. Gilbert, M. Heiner, and S. Lehrack. A Unifying Frameworks for Modelling and Analysing Biochemical Pathways Using Petri Nets. *Proceedings of the Workshop on Computational Methods in Systems Biology (CMSB)*, pages 200–216, 2007.
32. P. J. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc. Natl. Acad. Sci. USA*, 95(12):6750–6755, 1998.
33. M. Heiner, D. Gilbert, and R. Donaldson. Petri Nets for Systems and Synthetic Biology. In *Proc. of SFM'08*, volume 5016 of *LNCS*, pages 215–264. Springer, 2008.
34. M. Heiner, I. Koch, and S. Schuster. Using time-dependent Petri nets for the analysis of metabolic networks. In R. Hofestadt, K. Lautenbach, and M. Lange, editors, *Workshop Modellierung und Simulation Metabolischer Netzwerke*, Preprint No.10, pages 15–21. Faculty of Computer Science, Otto-von-Guericke University of Magdeburg, 2000.
35. M. Heiner, I. Koch, and K. Voss. Analysis and Simulation of Steady States in Metabolic Pathways with Petri nets. *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'01)*, pages 15–34, 2001.
36. R. Hofestädt. A Petri net application of metbolic processes. *Journal of System Analysis, Modelling and Simulation*, 16:113–122, 1994.
37. I. Kanaris, K. Moutselos, A. Chatziioannou, I. Maglogiannis, and F.N. Kolisis. Building in-silico pathway SBML models from heterogeneous sources. In *BioInformatics and BioEngineering (BIBE2008)*, pages 1–6. IEEE, 2008.
38. M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi. KEGG for linking genomes to life and the environment. *Nucleic Acids Research*, pages D480–D484, 2008.
39. I. Koch and M. Heiner. Petri nets. In B. H. Junker and F. Schreiber, editors, *Analysis of Biological Networks*, Book Series in Bioinformatics, pages 139–179. Wiley & Sons, 2008.
40. K.E. Lee, M.H. Jang, A. Rhie, C.T. Thong, S. Yang, and H.S. Park. Java DOM parsers to convert KGML into SBML and BioPAX common exchange formats. *Genomics & Informatics*, 8(2):94–96, 2010.
41. W. Marwan, A. Sujatha, and C. Starostzik. Reconstructing the regulatory network controlling commitment and sporulation in physarum polycephalum based on hierarchical Petri net modelling and simulation. *Journal of Theoretical Biology*, 236:349–365, 2005.

42. H. Matsuno, Y. Tanaka, H. Aoshima, A. Doi, M. Matsui, and S. Miyano. Biopathway representation and simulation on hybrid functional Petri net. *In Silico Biology*, 3(0032), 2003.
43. S. Miyano and H. Matsuno. How to model and simulate biological pathways with Petri Nets - a new challenge for system biology. In *International Conference on Applications and Theory of Petri Nets, Bologna, Italy*, 2004.
44. T. Murata. Petri Nets: Properties, Analysis, and Applications. *Proceedings of IEEE*, 77(4):541–580, 1989.
45. J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.
46. L. Popova-Zeugmann, M. Heiner, and I. Koch. Timed Petri Nets for modelling and analysis of biochemical networks. *Fundamenta Informaticae*, 67:149–162, 2005.
47. Harsha K. Rajasimha. PathMeld: A methodology for the unification of metabolic pathway databases. Master's thesis, Virginia Polytechnic Institute and State University, 2004.
48. V. N. Reddy. Modeling Biological Pathways: A Discrete Event Systems Approach. Master's thesis, The Universisty of Maryland, ISR-M.S. 1994-4, 1994.
49. V. N. Reddy, M.N. Liebman, and M.L. Mavrovouniotis. Qualitative Analysis of Biochemical Reaction Systems. *Comput. Biol. Med.*, 26(1):9–24, 1996.
50. V. N. Reddy, M. L. Mavrovouniotis, and M. N. Liebman. Petri net representations in metabolic pathways. In *ISMB93: First Int. Conf. on Intelligent Systems for Molecular Biology*, pages 328–336. AAAI press, 1993.
51. W. Reisig. *Petri Nets: An Introduction*. EACTS Monographs on Theoretical Computer Science. Springer Verlag, 1985.
52. L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg. The Database of Interacting Proteins: 2004 update. *Nucleic Acids Research*, 32(Database issue: D449-D451), 2004.
53. P.H. Starke and S. Roch. The Integrated Net Analyzer. *Humbolt University Berlin*, 1999. www.informatik.hu-berlin.de/ starke/ina.html.
54. K. Voss, M. Heiner, and I. Koch. Steady state analysis of metabolic pathways using Petri nets. *In Silico Biology*, 3(0031), 2003.

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.117-134

Pain Signaling - A Case Study of the Modular Petri Net Modeling Concept with Prospect to a Protein-Oriented Modeling Platform

Mary Ann Blätke[1], Sonja Meyer[1] and Wolfgang Marwan[1]

[1]Otto-von-Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
wolfgang.marwan@ovgu.de

Abstract. The construction of monolithic pathway models, as well as their coupling, curation and the integration of new data is arduous and inconvenient. The modular Petri net modeling concept we present here shows one way to manage these difficulties. In our concept, proteins are represented as functional units by Petri net submodels with a defined structure and connection interface, called modules. Each module integrates all publicly available information about its intramolecular changes and interactions with other molecules. Hence, a module corresponds to an interactive review written in a formalized language. This allows to intuitively understand the functionality of a protein. Modules of interacting proteins communicate through matching subnets, which renders the automatic generation of molecular networks possible. Here, we demonstrate the applicability and advantages of our concept on pain signaling. The molecular mechanisms involved in pain signaling are complex and poorly understood. To enhance our understanding of the mechanisms and to get an impression of the functional interactions among the involved pathways, we systematically build a model from modules of pain-relevant proteins. We also offer a prospect of a platform to organize approved curated modules in order to generate molecular networks. Hopefully, our concept helps bridging the gap between experimental bioscientists and theoretically oriented systems biologists.

Key words: Petri net, modular approach, pain signaling, molecular networks

1 Introduction

The modeling of large molecular networks in systems biology is a challenging process, as well as their steady curation and improvement. Our modular Petri net modeling concept described here, offers a way to handle these challenges by combing Petri net modeling with a modular approach. Modular approaches have already conquered the field of systems biology [1]. In the biological context, just monolithic pathways have been regarded as single entities up to now. In our concept, proteins are represented as functional units by a Petri net with a defined

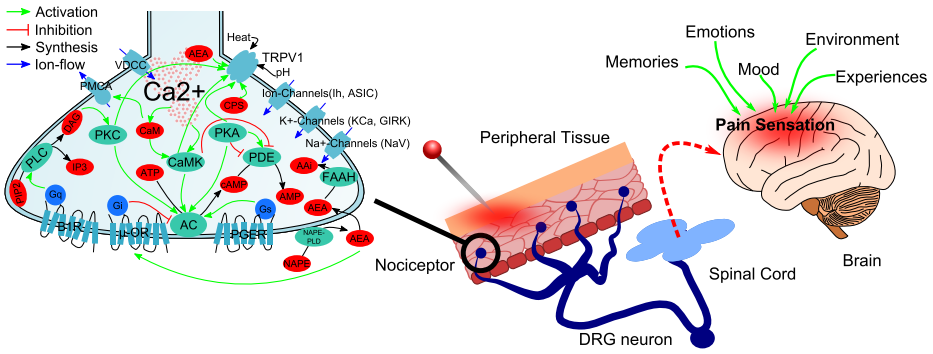


Figure 1: Nociceptor: Detector of noxious stimuli. The peripheral terminals of dorsal root ganglion (DRG) neurons, called nociceptors, detect noxious stimuli. Inside the nociceptor (left side) a plethora of signal cascades is responsible for the processing and the encoding of the noxious stimuli operating on membrane components. The action potential induced by sensitization runs along the DRG neuron to the spinal cord, where other nociceptive neurons receive the signal. The brain integrates the transmitted signal of the noxious stimuli and behavioral information (emotions, mood, memories etc.) and causes the painful sensation.

structure and connection interface, called module. Each module comprises and integrates scattered information about individual proteins, its intramolecular changes and interactions with other molecules. Therefore, a module is equivalent to an interactive review article written in a formalized language with the help of Petri nets. The graphical notation of the underlying mathematical model allows to intuitively understand the modeled protein. Modules of interacting proteins communicate through identical matching subnets, the connection interface. The coupling of monolithic pathway models is far from trivial in contrast to the combination of protein modules described here. The defined connection interface of each module allows to easily generate a comprehensive model of a molecular network from a set of modules. However, it has been proven that Petri nets are ideally suited to describe biological processes by their very nature. The Petri net formalism provides a mathematical language to describe parallel and concurrent processes of bipartite systems [2]. Therefore, we choose Petri nets to describe the molecular network of pain signaling (case study). Pain signaling comprises complex and diverse molecular mechanisms of parallel, convergent and concurrent processes. Up to now, a large variety of molecular pain mediators is known **Figure 1**. Nevertheless, especially the intracellular plethora of pain signaling cascades triggered by membrane components is underinvestigated and therefore partly unknown [3]. The challenge to represent pain signaling in terms of a comprehensive Petri net model has led to the development of our modular Petri net modeling concept and a modular network describing pain signaling in the peripheral terminals of dorsal root ganglion (DRG) neurons **Figure 1**. Our concept is not at all limited to pain signaling, the application to other molecular net-

works is straightforward. However, by studying pain signaling it turned out that our concept is well suited to handle large molecular networks. Since last year, we improved and extended our modular Petri net modeling concept presented here (compare [4]). In addition, we developed first ideas to manage the modules after curation by bioscientists in a database and thus, provide a platform to the scientific community. The platform also facilitates the automatic generation of a model of a molecular network from a collection of approved curated modules.

2 Petri Net Formalism

Petri nets offer a mathematical modeling language to describe concurrent and parallel processes, as well as communication and synchronization in bipartite systems. The graphical notation and construction of Petri nets allows to intuitively model such processes while being formally and mathematically consistent. Therefore, Petri nets are ideally suited to describe biological processes by their very nature [2]. The standard Petri net [2] consists of four elements: places, transitions, arcs and tokens. In biological systems places correspond to species (chemical compounds) and transitions describe the action occurring among the species ((bio-)chemical reactions). Arcs specify the relations between places and transitions. Tokens refer to the amount (discrete number, concentration) of a species. Further, transitions are allowed to fire (enabled) if all pre-places are sufficiently marked. By firing of transition it deletes tokens from its pre-places and produces tokens on its post-places.

Definition 1 (Petri net). [1] A Petri net *is a quadruple* $N = (P, T, f, m_0)$, *where:*

- P, T *are finite, non empty, disjoint sets.* P *is the set of* places. T *is the set of* transitions.
- f: $((P \times T) \cup (T \times P) \rightarrow \mathbb{N}_0$ *defines the set of directed* arcs, *weighted by non-negative integer values*
- m_0: $P \rightarrow \mathbb{N}_0$ *gives the* initial marking.

One benefit of Petri nets is the formal analysis of the network structure. The topological properties of a Petri net are also meaningful in a biological context and give valuable hints to validate the network structure [2]. Important criteria to validate a biological Petri net are liveness, boundedness, reversibility, as well as T- and P-Invariants [2].

Definition 2 (Boundedness). [1]

- *A place* p *is* k-bounded *if there exists a positive integer number* k, *which represents an upper bound for the number of tokens on this place in all reachable markings of the Petri net:*
  $\exists k \in \mathbb{N}_0 : \forall m \in [m_0\rangle : m(p) \leq k.$

- *A Petri net is* k-bounded *if all its places are* k-bounded.
- *A Petri net is* structurally bounded *if it is bounded in any initial marking.*

Definition 3 (Liveness). [1]

- *A transition* t *is* dead in the marking m *if it is not enabled in any marking* m' *reachable from:* $\nexists m' \in [m\rangle : m'(t)$.
- *A transition* t *is* live *if it is not dead in any marking reachable from* m_0.
- *A marking* m *is* dead *if there is no transitions which is enabled in* m.
- *A Petri net is* deadlock-free *if there are no reachable dead markings.*
- *A Petri net is* live *if each transitions is live.*

Definition 4 (Reversibility). [1] *A Petri net is* reversible *if the initial marking can be reached again from each reachable marking:* $\forall m \in [m_0\rangle : m_0 \in [m\rangle$.

Definition 5 (P-invariants, T-invariants). [1]

- *The* incidence matrix *of* N *is a matrix* $\mathbb{C} : P \times T \to \mathbb{Z}$*, indexed by* P *and* T*, such that* $\mathbb{C}(p,t) = f(t,p) - f(p-t)$.
- *A* place vector *(transition vector) is a vector* $x : P \to \mathbb{Z}$*, indexed by* P *(*$y : T \to \mathbb{Z}$*, indexed by* T*)*
- *A place vector (transition vector) is called* P-invariant (T-invariant) *if it is a nontrivial nonnegative integer solution of the linear equation system* $x \cdot \mathbb{C} = 0$ *(*$\mathbb{C} \cdot y = 0$*).*
- *The set of nodes corresponding to an invariant's nonzero entries are called the* support *of this invariant* x*, written as* $supp(x)$.
- *An invariant* x *is called* minimal *if* $\nexists$ *invariant* z*:* $supp(z) \subset supp(x)$*, i.e. its support does not contain the support of any other invariant* z*, and the greatest common divisor of all nonzero entries of* x *is 1.*
- *A net is* covered by P-Invariants (T-invariants) *if every place (transition) belongs to a P-invariant (T-invariant).*

Thereby, we can determine if the model of the molecular network contains dead-states, is live or if it can reset its initial state (reversible). To ensure the mass conversation the coverage by P-invariants and the boundedness of the Petri net must be considered. P-invariants describe sets of related species or states of a species. Boundedness ascertains that no species infinitely accumulates in the network. T-invariants contain a set of actions/reactions to reset its initial state [2].

Several specialized Petri net classes like qualitative, stochastic, continuous, hybrid Petri nets and their colored counterparts are available to describe different scenarios and to consider different simulative approaches. All network classes are convertible into each other without changing the network structure. This allows the application of the same powerful analysis techniques to the underlying qualitative structure for all Petri net network classes [2].

In particular, we use stochastic Petri nets to describe the inherently stochastic nature of biological processes. In addition to the standard Petri net, firing rates are assigned to each transition, which are determined by random variables depending on the probability distribution. Therefore, we use stochastic simulation to investigate the dynamic behavior by the time-dependent token flow [2].

Definition 6 (Stochastic Petri Net). [1] *A biochemically interpreted stochastic Petri net is a quintuple SPN_{Bio} = (P, T, f, v, m_0), where:*

- *P, T are finite, non empty, disjoint sets. P is the set of* places. *T is the set of* transitions.
- *f: $((P \times T) \cup (T \times P) \rightarrow \mathbb{N}_0$ defines the set of directed* arcs, *weighted by non-negative integer values*
- *v: $T \rightarrow H$ is a function, which assigns a stochastic hazard function h_t to each transition t, whereby* $H := \bigcup_{t \in T} \left\{ h_t | \, h_t : \mathbb{N}_0^{|^\bullet t|} \rightarrow \mathbb{N}^+ \right\}$ *is the set of all stochastic hazard functions, and v(t) = h_t for all transitions $t \in T$.*
- *m_0: $P \rightarrow \mathbb{N}_0$ gives the* initial marking.

3 Modular Modeling Concept

The enormous amount of regulative events in pain signaling **Figure 1** results into the development of a modular modeling concept, which considers every protein as functional independent unit. The basic concept presented last year (see reference [4]) has now been improved and extended to a defined modeling concept. The suggested modular modeling concept uses Petri nets to allow the assembling of molecular networks from functional Petri net submodels of the involved proteins with a defined structure and connection interface, called modules. A module reflects all the intramolecular changes of a protein and its interactions with other molecules as reported in the literature. Therefore, a module comprises wide-spread information about each protein. **Figure 3 and 4** show exemplary the modules of two proteins and their regulation. Non-proteins (ions, second messenger, energy equivalents etc.) are contained in the modules as interactants and indirectly connect the proteins. The structure and the dynamic behavior of each module has to fulfill certain criteria to be valid to meet the requirements of our modular Petri net modeling concept **Table 1**. After positive validation, the modules can be easily linked to a modular network by the defined connection interface of each module. Additionally, we are now able to predict the properties of the complex modular network from the topological properties of the combined modules **Table 1**. In this section, we explain all steps needed for the construction of a single module from literature and the assembling of the modular network from the constructed modules.

3.1 Network Structure and Properties of a Module

We construct modules based on the information about the structure of a protein, interactions with other components and intramolecular changes during its regulation given in the literature. Each place of a module corresponds to a specific state of a functional protein domain (phosphorylation site, catalytic and

[1] Mathematical definitions are taken from [2].

inhibitory domain etc.) or a specific state of a non-protein (free or bound, substrate or product etc.). In this context, a transition describes a shift between two different states of a protein domain or non-protein by a molecular action (binding/dissociation, (de-)phosphorylation, conformational changes, substrate processing etc.). Each module is constructed in such a way, that it obeys criteria important for biological networks, which are given in detail in [2] and summarized in **Table 1**. All states of a protein domain and all states of a non-protein constitute a P-invariant (see also **Section 2**). Therefore, the whole module must be covered by P-invariants. In this context, P-invariants ensure mass conservation. Sequential state shifts that restore an initial state of a protein domain or a non-protein form T-invariants (see also **Section 2**). Consequently, all T-invariants are covered by P-invariants. Places connected with a transition by a double arc (see **Figure 3 and 4**) indicate molecular states responsible for other state shifts without changing itself. To prohibit external sinks and sources of protein domains, the modules are not bounded by transitions. However, a module might well be bounded by places. Boundary places have their origin in modules of other proteins or represent non-proteins, which are consumed or produced. The principle of double entry-bookkeeping is a part of the modular modeling concept, since every module has to contain all interactions with other molecules. Thus, modules of two interacting proteins share identical matching subnets describing the interaction mechanism.

Table 1: Topological properties of the modules and their biological interpretation linked with their transferability to the modular network.

Properties	Module	Modular network
A Properties that must be fulfilled for each module		
Ordinary	Just natural elementary regulation steps are considered. Therefore, the arc weights are uniformly set to "1".	All properties are direct transferable to the modular network, because they are fulfilled by all modules.
Homogeneous	Due to ordinary: state-shifts produce (consume) the same amount of tokens on each place.	
Connected	Among all different states of protein domains and non-proteins exist at least one indirect path to represent the interrelated structure of a protein.	
Covered with P-invariants	A set of related states of a protein domain or of a non-protein must form a P-invariant. In consequence, the module must be covered with P-invariants.	
Boundedness	The coverage with P-invariants causes boundedness of each module and avoids the infinite accumulation of tokens in a module.	
B Properties that must not be fulfilled for each module		
Pure	Every module contains states of a protein domain responsible for other state shifts without changing itself. Therefore, each module contains double arcs.	

Boundary Transitions	The modules are not bounded by transitions to avoid external sinks and sources of protein domains and non-proteins.	All properties are direct transferable to the modular network, because they are fulfilled by all modules.
Conservative	The formation of protein complexes results in a non-token-preservingly firing of transitions.	
Static conflict free	A module contains at least one state of a protein domain or a non-protein attending on multiple state shifts.	
Strongly covered with T-Invariants	A module contains two sequential actions reproducing the initial state of the involved protein domain or non-protein.	
C Properties that are variable among all modules		
Dead Transition	Depending on the initial marking.	Depending on the initial marking.
Dead states	Depending on the specific regulation of a protein, the respective module contains at least one set of sequential state shifts acting independent of all other actions. In this case, the module has no dead state.	The modular network has no dead state if a least one module has no dead state.
Dynamic conflict free	Depending on the specific regulation of a protein, certain state shifts in the respective module do not inhibit other actions in the same module. Consequently, the module has no dynamic conflicts.	The modular network is not dynamic conflict free, if one module contains a dynamic conflict.
Boundary places	Depending on the specific regulation of a protein, the respective module contains places corresponding to protein domains of other interacting proteins or non-proteins that are irreversibly changed in the respective module. In this case, the module has boundary places. The following properties cannot be fulfilled if a module has at least one boundary place: – Strongly connected – Non-blocking multiplicity – Covered with T-Invariants – Siphon-Trap Property – Liveness	All of these properties can be transfered to the modular network, if at least one module in the modular network has at least one boundary place that does not gain a pretransition after module coupling.

3.2 Validation of a Module

The topological properties can be used for the validation of each module and locating inconsistencies in the module structure [2]. Each topological property has a significant biological interpretation. A set of those properties must be fulfilled, another set must not be fulfilled and a third set is variable depending on the unique function and structure of each module **Table 1** (see [2] for further explanations). After the construction of a module, its structure is checked whether it obeys the given criteria given in **Table 1**. Each module is also subjected to stochastic simulation studies. The kinetic function of each transition

can be defined by known kinetic parameters (binding, dissociations, and affinity constants) or more complex kinetic equations (michaelis menten, hill kinetic). If kinetic information are not available, the kinetic parameters can be determined by trial and error or by more sophisticated parameter estimation methods. The observed dynamic behavior, i.e. the time-dependent token-flow (see also references [4,5,6]), must in principle reflect the modeled effector function. A module is valid if its structure confirms the given topological properties and the dynamic behavior reflects the experimental obtained time curves.

3.3 Generation of a Modular Network

Next, the modules are connected by identical places of non-proteins and identical matching subnets among the modules. All shared elements have to be indicated as such in each module by declaring the included transitions and places as logical nodes (connection interface). Afterwards, the modules can be combined to one comprehensive simulative model. No further interventions are required. The coupling procedure does not affect the structure and properties of each module. Even the kinetic of the modules are kept and inherited to the resulting modular network. Hence, simulation with the modular network can be performed right after its generation.

3.4 Properties of the Modular Network

A new important achievement of the modular modeling concept is the determination of properties of the complex network from submodels, which might also be interesting for other Petri net applications. Due to the defined structure, the resulting uniform topological properties and defined connection interface of the modules, we are able to predict the properties of the modular network. Obviously, the respective non-variable properties among the modules can be transferred one-on-one to the modular network **Table 1**. From the comparison of the fulfillment of each variable property among the modules it can be deduced, whether the respective property holds for the modular network (see also **Table 1** for more details). Therefore, all properties of the modular network are derivable from the respective set of modules that assemble the modular network. Computational analyses with the place/transition analyzer Charlie [7] confirm the predicted properties (not shown here, see [6] for more details).

4 A Model Relevant for Pain Signaling

Pain signaling comprises complex and diverse processes. Therefore, a plethora of proteins and other components participate in the molecular regulation of painful sensations (see also **Figure 1**). Several members of the G-protein-coupled receptor family (GPCRs) are involved in pain signaling like opioid, cannabinoid, muscarinic, prostaglandine and β-2-adrenergic receptors. The GPCRs act through their G-proteins on adenylyl cyclases (Type VIII, V, I), phospholipase Cβ and

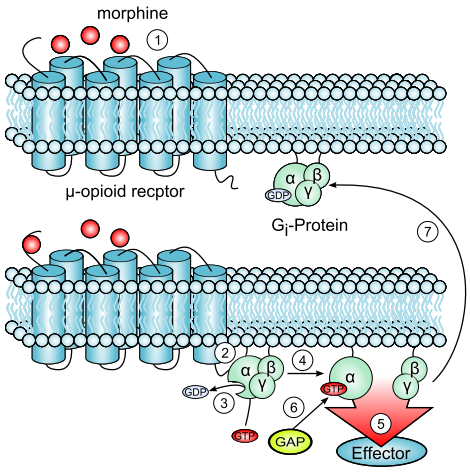


Figure 2: Activation of the Gi-protein by μ-opioid receptor. Morphine binds to the extracellular site of the μ-opioid receptor (muOR) **1**. Therefore, muOR changes its conformation and binds the Gi-protein **2**. GDP is exchanged by GTP at the Giα-subunit which in turn activates the Gi-protein **3**. The active Gi-protein dissociates into the Giα and Giβ/γ-subunit **4**. The Gi-protein subunits can now interact with their downstream targets **5**. GTP is hydrolyzed by the intrinsic GTPase of the Giα-subunit due to stimulation by a GTPase activating protein (GAP) **6**. The G-protein subunits reassociate if the Giα-subunit is inactive **7**. **Figure 3 and 4** show these mechanisms translated into two respective modules of muOR and the Gi-Protein.

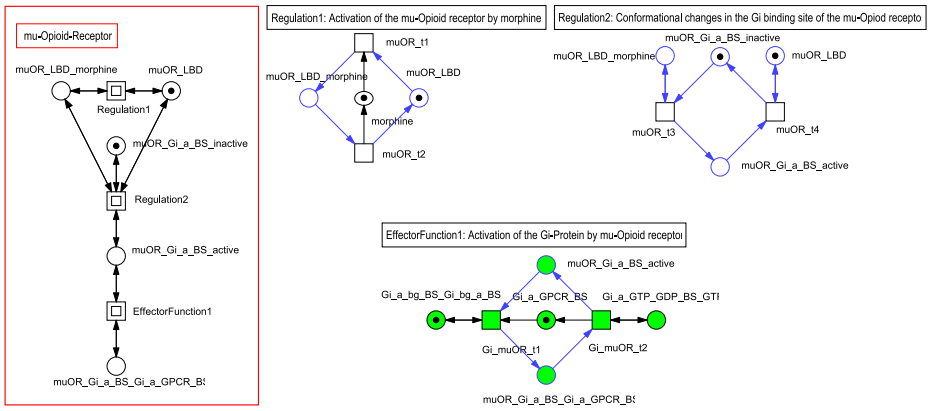


Figure 3: Module of the μ-opioid receptor. The module represents the regulation of the μ-opioid receptor as shown in **Figure 2**. The top-level of the μ-opioid receptor module is shown in the red rectangle on the left site. The macro transitions (black boxed squares) contain subnets on a deeper level. The corresponding subnets are shown on the right site and below. The blue arcs and blue framed places mark inputs from places shown on the top level. Green filled nodes indicate the connections interface that is used to connect the module of the μ-opioid receptor with the module of the Gi-protein **Figure 4**. Actions of the respective transitions are given in **Table 2**.

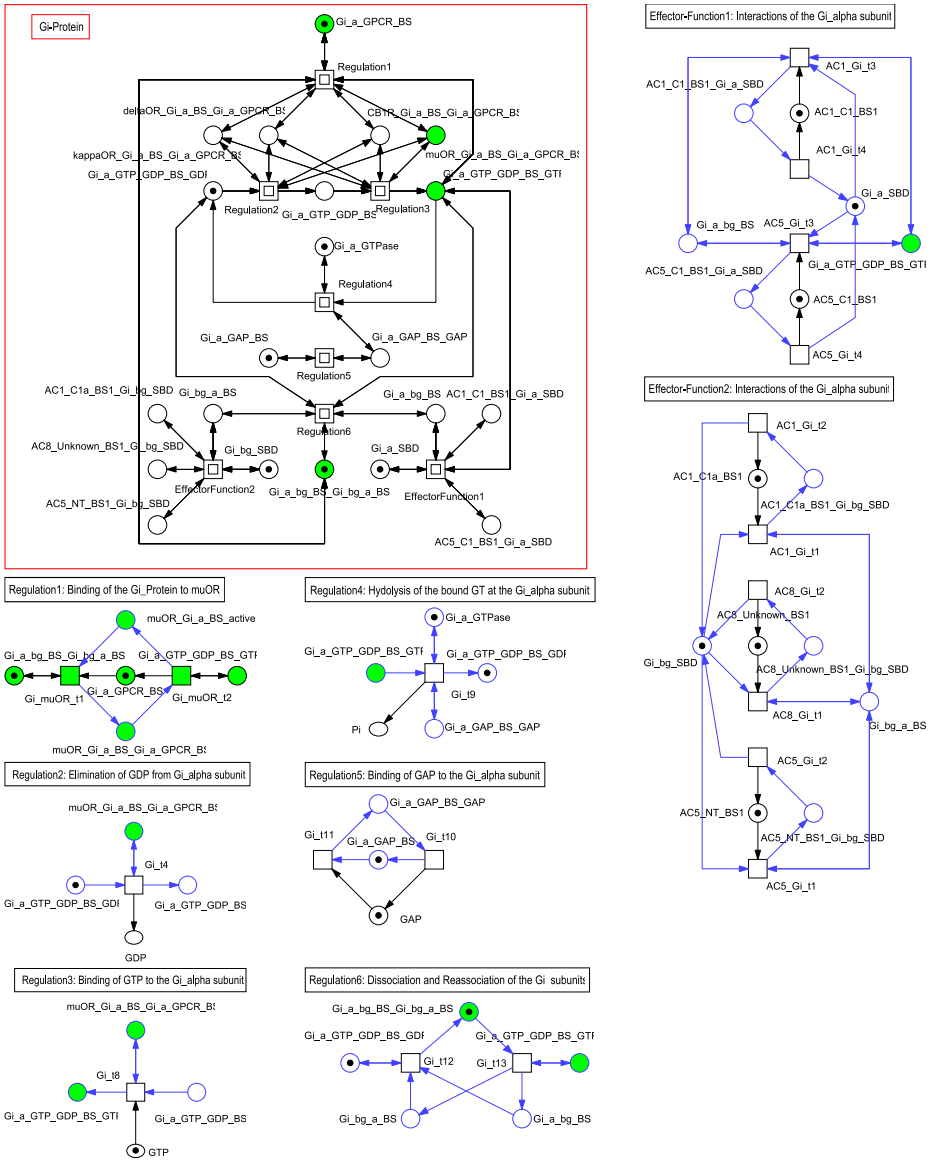


Figure 4: Module of the Gi-protein. The module represents the regulation of the Gi-protein as shown in **Figure 2**. The top-level of the Gi-protein is shown in the red rectangle on the left site. The macro transitions (black boxed squares) contain subnets on a deeper level. The corresponding subnets are shown on the right site and below. The blue arcs and blue framed places mark inputs from places shown on the top level. Green filled nodes indicate the connections interface that is used to connect the module of Gi-protein with the module of the μ-opioid receptor **Figure 3**. Actions of the respective transitions are given in **Table 2**.

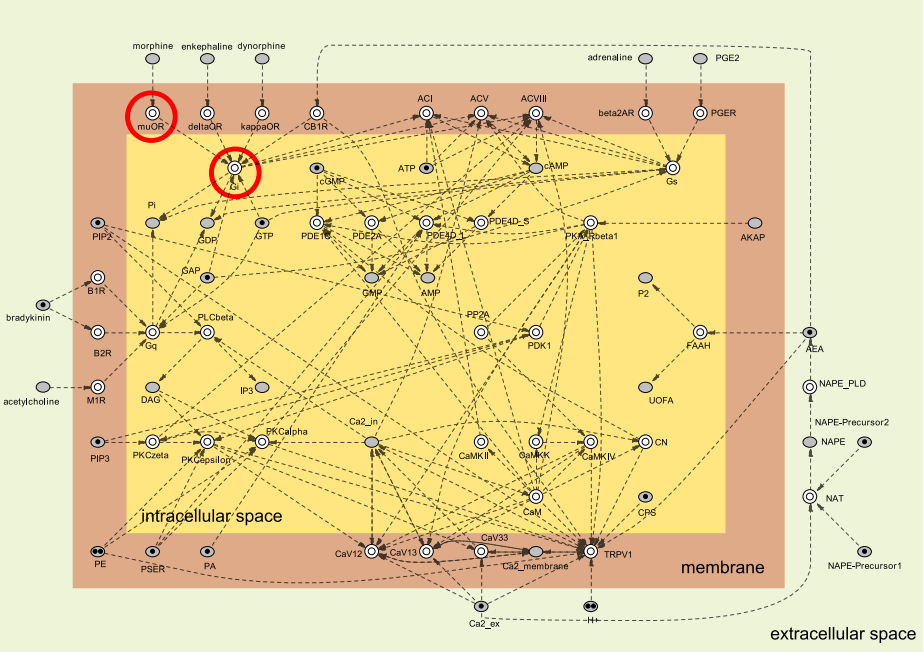


Figure 5: Modular Network Explaining Molecular Pain Mechanisms. Each macro place (boxed circles) contains a module of a pain-related protein. **Figure 3 and 4** (red circles) show two examples in detail. The grey shaded (logical) places represent the involved non-proteins (exceptions AKAP and GAP). The modules are arranged according to their localization (intracellular, membrane components, extracellular). The dashed arcs (no Petri net elements) indicate the interactions among all pain-related molecules.

ion-channels. Numerous protein kinases regulate pain signaling among them are different isoforms of PKA (RIIβ), PKC (α, ϵ, ζ), CaMK (II, IV). Opponents of the protein kinases are protein phosphatase 2A and calcineurin. Calmodulin is an important calcium-binding protein interacting with other pain-related protein like the voltage dependent calcium channels (CaV1.2, CaV1.3, CaV3.3) and the capsaicin receptor (TRPV1). Second messenger like DAG, Ca^{2+} and cAMP are also important components in the context of pain signaling and indirectly link proteins [3].

Each of those pain-relevant proteins is represented by its respective module. In total, 38 modules of pain-relevant proteins have been derived from clinical pain literature (all references can be found in [6]). New modules have been added and old modules have been updated by formulating the mechanisms in more detail since last year (e.g. compare the modules in **Figure 3 and 4** with the respective module shown in [4]). Exemplary, we show the mechanisms of the Gi-protein activation by the μ-opioid receptor **Figure 2**. Both, the μ-opioid receptor and the Gi-protein are represented by functional connectable modules (see **Figure**

3 and 4). The biological meaning of each transition refering to the steps shown in **Figure 2** are given in **Table 2**.

Table 2: Biological interpretation of transitions contained in the modules of the μ-opioid receptor and Gi-Protein (compare **Figure 3 and 4**)

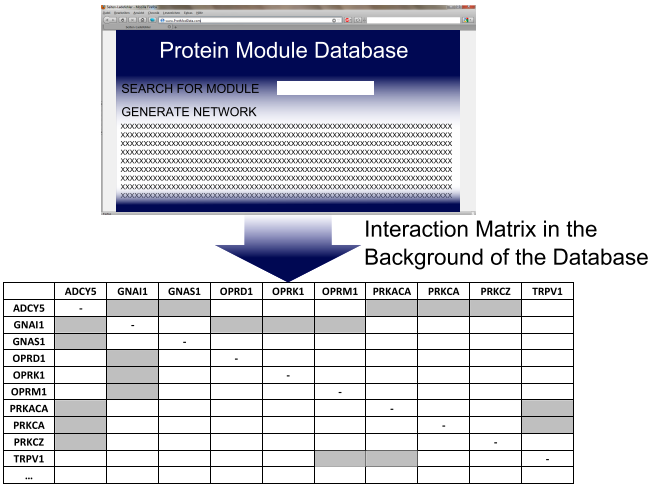
muOR_t1	- Binding of morphine to the ligand binding site of muOR
muOR_t2	- Dissociation of morphine from the ligand binding site of muOR
muOR_t3	- Conformational changes in the $G_i\alpha$ binding site of muOR due to the bound morphine (= activation)
muOR_t4	- Reverse conformational changes of the $G_i\alpha$ binding site of muOR if morphine is not bound to muOR (= inactivation)
G_i_muOR_t1	- Binding of the $G_i\alpha$ subunit (in complex with $G_i\beta/\gamma$) to the active $G_i\alpha$ binding site of muOR
G_i_muOR_t2	- Dissociation of the $G_i\alpha$ subunit from the active $G_i\alpha$ binding site of muOR if GDP is exchanged by GTP at the $G_i\alpha$ subunit
G_i_t1	- Dissociation of GDP from the GTP/GDP binding site of $G_i\alpha$ if $G_i\alpha$ is bound to muOR
G_i_t2	- Binding of GTP to the GTP/GDP binding domain of $G_i\alpha$ if $G_i\alpha$ is still bound to muOR
G_i_t3	- Hydrolysis of the bound GTP to GDP by the GPTase domain of $G_i\alpha$ if GAP is bound to $G_i\alpha$
G_i_t4	- Dissociation of the $G_i\alpha$-$G_i\beta/\gamma$ complex if GTP if bound to $G_i\alpha$
G_i_t5	- Reassociation of the $G_i\alpha$ and $G_i\beta/\gamma$ subunits if the $G_i\alpha$ subunit is loaded with GDP
G_i_t6	- Binding of GAP to the GAP binding site of $G_i\alpha$
G_i_t7	- Dissociation of GAP from the GAP binding site of $G_i\alpha$
AC8_G_i_t1	- Binding of $G_i\beta/\gamma$ subunit to AC8 at an unknown binding domian
AC8_G_i_t2	- Dissociation of $G_i\beta/\gamma$ subunit from AC8
AC8_G_i_t3	- Binding of $G_i\beta/\gamma$ subunit to AC1 at the C1a domain
AC8_G_i_t4	- Dissociation of $G_i\beta/\gamma$ subunit from AC1
AC8_G_i_t5	- Binding of $G_i\alpha$ subunit to AC1 at the C1 domain
AC8_G_i_t6	- Dissociation of $G_i\alpha$ subunit from AC1
AC8_G_i_t7	- Binding of $G_i\alpha$ subunit to AC5 at the C1 domain
AC8_G_i_t8	- Dissociation of $G_i\alpha$ subunit from AC5

The resulting modular network **Figure 5** generated from the set of modules of pain-relevant proteins consists of 713 places and 775 transitions spread over 325 pages with a nesting depth of 4. The top level of the modular network shown in **Figure 5** contains all non-proteins (logic places in grey) and modules of pain-relevant proteins represented by single macro places (boxed circles). **Figure 3 and 4** depict two of these modules exemplarily. All components are arranged by their localization in the nociceptor. Due to the complexity of the regulation events, the modules are hierarchically designed to conserve the neat-arrangement offered by Petri nets. The modules communicate through identical matching subnets among them on lower levels and non-proteins. Therefore, the interaction among the displayed components are not visible on the top level of the modular network in **Figure 5**.

Here, we added arcs (no Petri net element) to the top-level shown in **Figure 5** to indicate the interactions among the components involved in pain signaling. **Figure 5** illustrates the high degree of interactivity among the components which was not obvious from the literature. The authors of reference [3] discuss whether the pathways involved in pain signaling are parallel or convergent. The interaction scheme in **Figure 5** clearly indicates that the involved pathways are highly convergent and influence each other. Several feedback loops are contained (not shown here) to regulate the cAMP- and Ca^{2+}- level and the membrane voltage, which are important for the sensitization of the nociceptor and the initiation of action potentials resulting into painful sensations. Therefore, the regulation of pain signaling is very complex and the resulting dynamic behavior is not trivial at all. The modules of the pain-relevant proteins are still in the curation process by the pain community. Since kinetic data is still rare in the pain signaling context, we have not been yet able to parameterize the modules and therefore to perform reliable simulation studies. The topological properties of the achieved modular network confirm the predicted properties of a common modular network given in **Table 2**.

5 Work in Progress: Establishing a Protein-Module Platform

We developed first basic ideas of a new protein-orientated modeling platform to open our modular Petri net modeling concept and the modules to the scientific community. This platform and the modular Petri net modeling concept provide the framework to organize the connectable modules in a database and to generate computational models of molecular networks from a central collection of approved curated modules.
Additionally to the Petri net of each module, a dataset will be provided in our database to characterize each module and the represented protein. The dataset contains information about the author and curator, the names of all places and transitions and their meaning, references to relevant literature used for the construction of the module, a list of open issues and information about the protein (accession number, gene symbol, synonyms, taxonomic classification, involved pathways etc.) extracted from the UniProt database [8].
The strict obedience of a naming convention for each node is the most important prerequisite to correctly link the modules by identical matching subnets and logical places of non-proteins. The identity of each module is determined by the unique accession number for proteins provided by UniProt [8]. The accession number will be used as prefix for all nodes but is not shown in the graph. More readable gene symbols for each protein, which are also provided from UniProt [8], are used as nicknames combined with common abbreviations of domains and their states to define a unique name for each place. The unique name of a transition is created from the gene symbols of the involved proteins and a counter. As a consequence, the module coupling is insensitive against author and version of a module, but sensitive against the organism. Also, the name of each node can



	ADCY5	GNAI1	GNAS1	OPRD1	OPRK1	OPRM1	PRKACA	PRKCA	PRKCZ	TRPV1
ADCY5	-									
GNAI1		-								
GNAS1			-							
OPRD1				-						
OPRK1					-					
OPRM1						-				
PRKACA							-			
PRKCA								-		
PRKCZ									-	
TRPV1										-
...										

Figure 6: Public Protein Module Platform. The protein-module platform allows to organize the modules and opens our modular Petri net modeling concept as well as the constructed modules to the scientific community. Based on shared nodes among the modules, an interaction matrix is created to control the module coupling (shown here on pain signaling).

be automatically generated. Checks will be integrated to prove the correctness of the names and thereby the connectivity of the module to its interactants.
To control the generation of a network from modules an interaction matrix is provided **Figure 6**. An interaction matrix, derived from the places of all modules, indicates which modules can be coupled by a common set of nodes. Thus, it controls the module coupling. First of all, the user adjusts the stringency at the organism level. Based on the interaction matrix, the generation of a modular network from modules might now occur in two ways: (a) pathway-oriented suggestion of a set interacting proteins, (b) iterative search of interactants from a chosen start protein. The pathway-oriented generation of modular networks will be achieved by tags that are added to the dataset of each module referring to involved pathways (e.g. pain signaling) and localization (e.g. DRG neuron, nociceptor). In a next step the connectivity of the modules has to be proven by the interaction matrix. **Figure 7** illustrates the application of the iterative search algorithm to generate a submodel of the model shown in **Figure 5**. Both possibilities lead to a list of interacting proteins showing all suitable modules. The user chooses for every protein the preferred module, if different versions of a module for one protein exist. Based on the module selection a comprehensive modular network is generated. The user can now execute simulations with the model and apply advanced structural analysis methods to investigate the model of the molecular network.

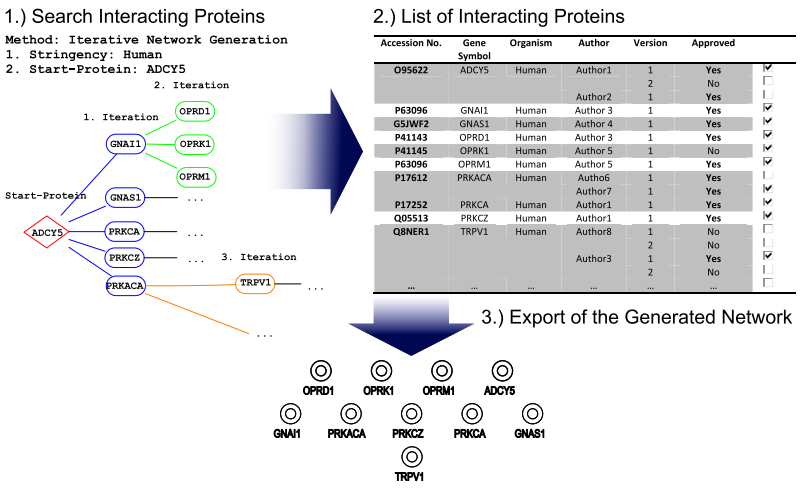


Figure 7: Iterative Generation of a Modular Network Before generating a modular network, the stringency is set by the user. A Start-protein must be chosen if the iterative search is applied to generate a network. Based on the interaction matrix **Figure 6**, the algorithm suggests new interaction partners step by step. After stopping the search, a list of all chosen proteins is created. Due to different opinions, there might exist different modules for one protein. In those cases, the user can choose the preferred module. Next, the modular network is generated from the chosen set of modules and exported.

6 Conclusion

The model relevant for pain signaling integrates the knowledge of approximately 320 scientific articles within 38 valid modules of important molecular pain actors in the nociceptor. The application of the modular modeling concept to the complex network of pain signaling proves its ability to cope with the specific demands of large and complex molecular networks. Our experience with biologists confirm the need of a molecule-oriented modeling concept. So far, the explained modular modeling concept is appreciated by our cooperation partners. It supports their work and improves the modeling of molecular networks. The concept is suited to test different hypothesis by exchanging different versions of a module in the modular network. Thus, the approach shed new light on molecular mechanisms. In a next straightforward step, the modules and therewith the complete modular network can be parameterized by experimental data, which are at the moment unfortunately still rare. After parameterization the model will be investigated by *in silico* experiments. The analysis of effects of systematic perturbation on the dynamic behavior of the model will help to pinpoint promising targets for the pain therapy. The extension of the model to colored Petri nets [9] enables the consideration of multiple copies of the pain-related proteins and of DRG neuron populations. The application of hybrid Petri nets [10] allows the combination of the current model with continuous models describing the generation of action

potentials in neurons. Sophisticated structural analysis methods will be applied to the model to screen for non-obvious properties that are defined by the Petri net structure. By this, we expect new information about multiple steady states, bifurcations and feedback loops that mainly determine the dynamic behavior of a network. At least, the validated and parameterized model and the mentioned investigations should contribute to the development of a mechanism-based pain therapy.

The modular modeling concept as such offers a lot of promising advantages and opportunities. All constructed modules can be easily reused in any other biological systems. Hence, the extension of the modular modeling concept to other biological systems is worthwhile. Every module by itself pools the currently spread knowledge about a protein and its interactants. The process of translating information about a protein into a module reveals missing interrelations. The modular modeling concept avoids inconsistencies in the entire complex modular network by constructing and validating first small independent submodels. The coupling procedure of the modules to an entire modular network by natural matching nodes is effortless. For different reasons, monolithic pathway models organized for example in the BioModels database [11] cannot be easily updated and combined with each other to give a more comprehensive model. Advantageously, the properties of the modular network can be deduced from the defined properties of the modules. The modeler and the curator just need to concentrate on one protein and its interactants. Also, the user has not inevitably to deal with the whole pathway and the theoretical concept itself.

The Petri net formalism itself offers quite few advantages against ODE models. As mentioned before, qualitative, continuous, stochastic and hybrid Petri nets as well as their colored counterparts are convertible in to each other without changing the qualitative structure. ODE systems do not offer the possibility to consider a model from such a range of corresponding sites without reconstructing the set of equations. Due to the graphical visualization of molecular networks by Petri nets, a bioscientist can intuitively understand the modeled mechanisms. This does not count for the mathematical representation of ODE systems. In case of ODE systems, the user has to deal with three different representations of a molecular network which do not obviously correspond to each other: (a) structure of the biological network, (b) the mathematical equations and (c) the implementation of those. Besides, the transformation of ODE systems into Petri nets is not unique. Various Petri nets can be constructed based on an ODE system [12]. The compact mathematical structure of an ODE might hide important biological information. Structural analysis techniques are sensitive to the respective structure of a Petri net. Therefore, the application of those techniques to Petri nets obtained from the variable transformation of ODEs leads also to variable results, which have to be treated with care [12].

The modular principle of the modeling concept offers some more possibilities to apply and extend the concept. The generated modular core network can be extended by gene expression, degradation and translocation modules. Even homo- and hetero-multimeric protein complexes can be modeled in detail with an ex-

tension modular modeling concept. Further, we plan to couple the network reconstruction for Petri nets [13] with the modular modeling concept. Here, nodes of the reconstructed network can be matched with corresponding modules. In addition, the minimal causal Petri nets reconstructed from experimental time series are extended with modules of the involved proteins.
The establishment of a platform for protein-modules and the opportunity to generate models of molecular networks form approved curated modules supports the switch from monolithic modeling to modular modeling of biological systems. Such a platform simplifies the exchange of data and knowledge among bioscientists by concentrating biological information about proteins and their interactants in the structure of the modules. Thereby, easing the access to systems biology for wetlab bioscientist.

7 Acknowledgement

This work is supported by the "Modeling Pain Switches" (MOPS) program of Federal Ministry of Education and Research (Funding Number: 0315449F). We thank Monika Heiner and co-workers for the outstanding cooperation on Petri nets and the software supply of Snoopy [14] and Charlie [7]. Further we thank the members of the MOPS consortia for supporting collaborations.

8 References

1. Alberghina, L, et al. A Modular Systems Biology Analysis of Cell Cycle Entrance into S-Phase. Systems Biology - Topic in Current Genetics. 2005, Vol. 13.
2. Heiner, M, Gilbert, D and Donaldson, R. Petri Nets for Systems and Synthetic Biology. [book auth.] M Bernardo, P Degano and G Zavattaro. SFM 2008. s.l. : Springer LNCS 5016. 2008.
3. Hucho, T and Levine, J. Signaling Pathway in Sensitization: Toward a Nociceptor Cell Biology. Neuron. 2007, Vol. 55.
4. Blätke, M.-A. et al. Petri Net Modeling via a Modular and Hierarchical Approach Applied to Nociception. Int.Workshop on Biological Processes & Petri Nets (BioPPN), satellite event of Petri Nets 2010. Braga, Portugal, 2010.
5. Blätke, M.-A., Marwan W. Modular and Hierarchical Modeling Concept for Large Biological Petri Nets Applied to Nociception.German Workshop on Algorithms and Tools for Petri Nets. Cottbus, Germany, 2010.
6. Blätke, M.-A. Petri-Netz Modellierung mittels eines modularen and hierarchischen Ansatzes mit Anwendung auf nozizeptive Signalkomponenten. Otto von Guericke University Magdeburg. 2010 (Diploma thesis).
7. Franzke, A. Charlie 2.0 - A Multi-Threaded Petri Net Analyzer. Brandenburg University of Technology Cottbus. 2009 (Diploma thesis).
8. Jain, E et al. Infrastructure for the Life Sciences: Design and Implementation of the UniProt Website. BMC Bioinformatics. 2009, Vol. 10.

9. Liu, F., Heiner, M. Colored Petri Nets to Model and Simulate Biological Systems. Int.Workshop on Biological Processes & Petri Nets (BioPPN), satellite event of Petri Nets 2010. Braga, Portugal, 2010.
10. Herajy, M., Heiner M. Hybrid Peri Nets for Modeling Hybrid Biochemical Interactions. German Workshop on Algorithms and Tools for Petri Nets. Cottbus, Germany, 2010.
11. Li, C. et al. BioModels Database: An Enhanced, Curated and Annotated Resource for Published Quantitative Kinetic Models. BMC Systems Biology. 2010, Vol. 4.
12. Soliman, S and Heiner, M. A Unique Transformation from Ordinary Differential Equations to Reaction Networks. PLoS ONE. 5, 2010.
13. Marwan, W., Wagler, A., Weismantel, R. Petri Nets as a Framework for the Reconstruction and Analysis of Signal Transduction Pathways and Regulatory Networks. Natural Computing. 2009.
14. Rohr, C, Marwan, W and Heiner, M. Snoopy - A Unifying Petri Net Framework to Investigate Biomolecular Networks. Bioinformatics. 2010, Vol. 26.

Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)
online: http://ceur-ws.org/Vol-724 pp.135-150

Multi-cell Modelling Using Coloured Petri Nets Applied to Planar Cell Polarity

Qian Gao[1], Fei Liu[3], David Tree[2], and David Gilbert[1]

[1] School of Information Systems, Computing and Maths, Brunel University, UK
http://www.brunel.ac.uk/siscm
[2] School of Health Science and Social Care, Brunel University, UK
http://www.brunel.ac.uk/about/acad/health/healthsub/bioscience
{qian.gao,david.tree,david.gilbert}@brunel.ac.uk
[3] Computer Science Department, Brandenburg University of Technology Cottbus, Germany
http://www-dssz.informatik.tu-cottbus.de/DSSZ
liu@Informatik.tu-Cottbus.de

Abstract. Modelling across multiple scales is a current challenge in Systems Biology. In this paper we present an approach to model at different spatial scales, applied to a tissue comprising multiple hexagonally packed cells in a honeycomb formation in order to describe the phenomenon of Planar Cell Polarity (PCP).
PCP occurs in the epithelia of many animals and can lead to the alignment of hairs and bristles. Here, we present an approach to model this phenomenon by applying coloured Petri Nets (CPN). The aim is to discover the basic principles of implementing CPN to model a multi-cellular system with a hierarchical structure while keeping the model mathematically tractable. We describe a method to represent a spatially defined multi-scale biological system in an abstract form as a CPN model, in which all reactions within a cell are categorised into two main types, each cell is sub-divided into seven logical compartments and adjacent cells are coupled via the formation of intercellular complexes. This work illustrates the issues that need to be considered when modelling a multi-cellular system using CPNs. Moreover, we illustrate different levels of abstraction that can be used in order to simplify such a complex system and perform sophisticated high level analysis. Some preliminary analysis results from animation and stochastic simulation are included in this paper to demonstrate what kinds of sequential analysis can be performed over a CPN model.

Keywords: coloured Petri nets, modelling, planar cell polarity

1 Introduction

With the rapid growth of data being generated in the biological field, it has become necessary to organise the data into coherent models that describe system behaviour, which are subsequently used for simulation, analysis or prediction.

Modelling plays a crucial role in facilitating the understanding of complex biological mechanisms from an holistic viewpoint.

A large variety of modelling approaches have already been applied to model a wide array of biological systems (see [1] for a review). Among them, Petri nets are particularly suitable for representing and modelling the concurrent, asynchronous and dynamic behaviour of biological systems. Since Reddy et al. [2] introduced the application of qualitative Petri nets to modelling of metabolic pathways, a large variety of applications of Petri nets (e.g. stochastic Petri nets, timed Petri nets, continuous Petri nets, and hybrid Petri nets, etc.) have been developed for modelling and simulating different types of biological systems [3], [4].

Modelling across multiple scales is a current challenge in Systems Biology. There is the need to model at different spatial scales to describe, for example, intracellular locality in compartments, organelles and vacuoles, as well as intercellular locality in terms of intercellular communication by complex formation across cell gaps, and by cytokines (intercellular messengers), and higher levels of organisation into tissues and organs composed of many cells. However, standard Petri nets do not readily scale to meet these challenges, and current attempts to simulate biological systems by standard Petri nets have been mainly restricted so far to relatively small models. Standard Petri nets tend to grow quickly for modelling complex systems, which makes it more difficult to manage and understand the nets, thus increasing the risk of modelling errors. Two known modelling concepts improving this situation are hierarchy and colour. Hierarchical structuring has been discussed a lot, e.g. in [5], while the colour has gained little attention so far.

While there is a lot of reported work on the application of different classes of standard Petri nets to a variety of biochemical networks, see [4] for a recent review, there are only a few which take advantage of the additional power and ease of modelling offered by coloured Petri nets (CPN). To our knowledge, the existing applications of CPN in systems biology can only be seen in [6], [7], [8], [9], [10], [11], [12], [13]. Moreover, these existing studies usually resort to `Design/CPN` [14] or its successor `CPN Tools` [15] in order to model and analyse biological systems. However neither tool was specifically designed with the requirements of Systems Biology in mind. Thus they are not suitable in many aspects, e.g. they do not directly support stochastic or continuous modelling, nor the simulative analysis of the models by stochastic or deterministic simulation.

Building upon the lessons learnt so far, we extend our software tool Snoopy [16], [17] by specific functionalities and features to support editing, simulating and analysing of biological models based on coloured qualitative, stochastic and continuous Petri nets. By doing so, we not only provide compact and readable representations of complex biological systems, but also do not lose the analysis capabilities of standard Petri nets, which can still be supported by automatic unfolding. Moreover, another attractive advantage of CPN for a biological modeller is that they provide the possibility to easily increase the size of a model consisting of many similar subnets just by adding new colours.

Modelling in Biology tends to emphasise molecular details. Yet in biological networks that involve more than a few components the typical situation is that many details are unknown, and it is imperative to devise an approach that can be insightful and predictive even in the absence of complete knowledge. Our strategy was based on building an abstract model of PCP which attempts to identify the key biological aspects (e.g. formation of intercellular complexes), and then constructing a more detailed but simple model which parameterises the many unknowns.

In this paper, our aim is to use CPN to describe an intercellular and intracellular signalling model that replicates the phenomenon of PCP in Drosophila wing. The epithelial cells in this organ are hexagonally packed in a 2-dimensional honeycomb lattice. The model incorporates an abstract description of information flow within the cell, and a representation of inter-cellular communication through the formation of protein complexes, so that local (transmembrane) signalling produces a global effect over the entire organ. It should be noted that approach presented in this paper is applied to an abstract model of PCP in order to illustrate the application of CPN to PCP signalling. Specifically, we focus on the way to include the honeycomb structure and logical compartments into the construction of our multi-cellular model.

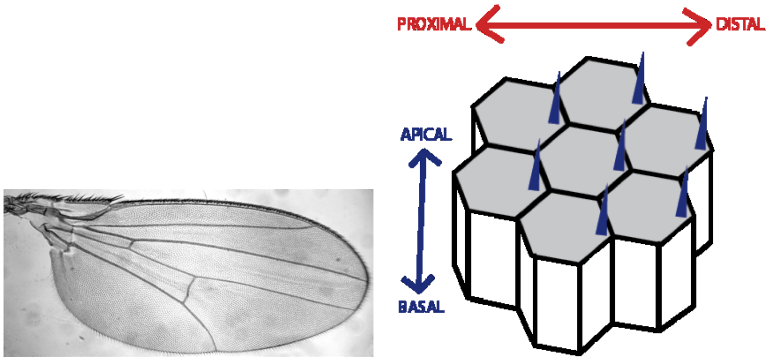
This paper is structured as follows: in Subsection 1.1 we introduced the biological background of planar cell polarity, followed by Subsection 1.2 briefly describing coloured Petri Nets and Section 2 describing our model and approaches, followed by the conclusion.

1.1 Planar Cell Polarity

Planar cell polarity (PCP) refers to the orientation of cells within the plane of the epithelium, orthogonal to the apical-basal polarity of the cells. This polarisation is required for many developmental events in both vertebrates and non-vertebrates. Defects in PCP in vertebrates underlie developmental abnormalities in multiple tissues including the neural tube, the kidney and the inner ear (reviewed in [18]). The signalling mechanisms underlying PCP have been studied most extensively in the epithelia of the fruit fly *Drosophila melanogaster* including the wing, the abdomen, the eye, and the bristles of the thorax. Genetic studies in the wing and eye in the 1990s led to the proposal of a PCP signalling pathway involving the PCP proteins Frizzled (Fz), Dishevelled (Dsh) and Prickle (Pk) (reviewed in [19]). In the late 1990s and 2000 further genetic analysis, including the discovery of more PCP proteins, e.g. Flamingo (Fmi) and Van-Gogh (Vang), and data on the sub-cellular localisation of these proteins in normal and mutant situations, have led to the formulation of more complex models of PCP signalling. In this paper we apply CPN to the models formulated in the fly wing as a means to gain insight into mechanism of PCP.

The adult Drosophila wing comprises about 300,000 hexagonal cells each of which contain a single hair which points in an invariant distal direction, see Figure 1. This hair comprises actin bundles and is extruded from the membrane at the distal edge of the cell during pupal development at the conclusion of

PCP signalling. Preceding this ultimate manifestation of PCP, PCP signalling occurs such that the proteins adopt an asymmetric localisation within each cell. At the initiation of PCP signalling Fmi, Fz, Dsh, Vang and Pk are all present symmetrically at the cell membrane. At the conclusion of PCP signalling Fmi is found at both the proximal and distal cell membrane, Fz and Dsh are found exclusively at the distal cell membrane and Vang and Pk are found exclusively at the proximal cell membrane. Through the interpretation of various genetic experiments a consensus view of the signalling events has been formulated that centres on the communication between these proteins at cell boundaries. The distally localised Fmi, Fz and Dsh recruit Fmi, Vang and Pk to the proximal cell boundary and vice versa. Since the localisation of the distal and proximal proteins appear to be mutually exclusive a completely polarised arrangement of protein localisation results. The PCP proteins are thus thought to mediate the cell-cell communication that comprises PCP signalling and that they are involved in establishing the molecular asymmetry within and between cells which is subsequently transformed into the polarisation of the wing hairs (reviewed in [20]). The result is a polarisation of individual cells and local alignment of polarity between neighbouring cells.



(a) (b)

Fig. 1. Drosophila: (a) Whole wing; (b) Schematic of hexagonal cells with hairs

Systems biology and mathematical modelling have been applied to PCP signalling by Amonlirdviman et al. [21] (extended in [22]) and Le Garrec et al. [23] (applied to the Drosophila eye in [24]). Both models centre around the idea of amplification of polarity via asymmetric complex formation of the core proteins. Both models rely on numerical simulations in two dimensions for fields of hexagonal or approximately hexagonal cells. Therefore, they tend to be rather complex and do not lend themselves to mathematical analysis very easily. Furthermore, because of the lack of appropriate biological data, the feedback mechanisms in these models are mainly based on assumptions.

In this paper, we apply CPN to PCP signalling in a generic setting that encompasses a broad class of specific models, ranging from a single cell model to a multi-cellular model. To this end, we have developed an abstract model for the generation of PCP to investigate the signalling by implementing animation analysis and stochastic simulation analysis.

1.2 Coloured Petri Nets

Coloured Petri nets (CPN) [25], [26] are a discrete event modelling formalism combining the strengths of Petri nets with the expressive power of programming languages. Petri nets provide the graphical notation and constructions for modelling systems with concurrency, communication and synchronisation. The programming languages offer the constructions for the definition of data types, then used for creating compact models. This is the greatest advantage of CPN.

CPN consist, as do standard Petri nets, of places, transitions and arcs. In systems biology, places also represent species (chemical compounds) while transitions represent any kind of chemical reactions. Each place gets assigned a colour set and contains distinguishable coloured tokens. A distribution of coloured tokens on all places together constitutes a marking of a CPN. Each transition may have a guard, which is a Boolean expression over defined variables. The guard must be evaluated to true for the enabling of the transition if it is present. Each arc gets assigned an expression, which is a multiset type of the colour set of the connected place.

The variables associated with a transition consist of the variables in the guard of the transition and in the expressions of arcs connected to the transition. Before the expressions are evaluated, the variables must get values assigned with suitable data types, which is called binding [26]. A binding of a transition corresponds to a transition instance in the unfolded net. Enabling and firing of a transition instance are based on the evaluation of its guard and arc expressions. If the guard is evaluated to true and the preplaces have sufficient tokens, the transition instance is enabled and may fire. When a transition instance fires, it removes coloured tokens from its preplaces and adds tokens to its postplaces, i.e. it changes the current marking to a new reachable one. The colours of the tokens that are removed from preplaces and added to postplaces are decided by arc expressions. The set of markings reachable from the initial marking constitutes the state space of a given net. These reachable markings and instances of transitions between them constitute the reachability graph of the net.

Thus CPN has the ability to tackle the challenges arising from modelling biological systems beyond one spatial scale, for example, repetition of components, which is the need to describe multiple cells each of which has a similar definition; and organisation of components, which refers to how cells are organised into regular or irregular patterns over spatial networks in one, two or three dimensions.

In the following, we give the formal definition of CPN and briefly describe the tool for modelling CPN.

Definition In CPN, there are different types of expressions, arc expressions, guards and expressions for defining initial markings. An expression is built up from variables, constants, and operation symbols. It is not only associated with a particular colour set, but also written in terms of a predefined syntax. In the following, we denote by EXP a set of expressions that comply with a predefined syntax. The formal definition for coloured Petri nets is as follows [25], [26].

Definition 1 (coloured Petri net). *A coloured Petri net is a tuple* $N =< P, T, F, \sum, C, g, f, m_0 >$*, where:*

- P *is a finite, non-empty set of places.*
- T *is a finite, non-empty set of transitions.*
- F *is a finite set of directed arcs.*
- $\sum$ *is a finite, non-empty set of colour sets.*
- $C : P \rightarrow \sum$ *is a colour function that assigns to each place* $p \in P$ *a colour set* $C(p) \in \sum$*.*
- $g : T \rightarrow EXP$ *is a guard function that assigns to each transition* $t \in T$ *a guard expression of the Boolean type.*
- $f : F \rightarrow EXP$ *is an arc function that assigns to each arc* $a \in F$ *an arc expression of a multiset type* $C(p)_{MS}$*, where* p *is the place connected to the arc* a*.*
- $m_0 : P \rightarrow EXP$ *is an initialisation function that assigns to each place* $p \in P$ *an initialisation expression of a multiset type* $C(p)_{MS}$*.*

If we consider special arcs, e.g. read arcs or inhibitor arcs, we can get coloured qualitative (extended) Petri nets. If the transitions are associated with random (or deterministic) firing rates, we will get coloured stochastic (or continuous) Petri nets [17].

Modelling tool In Snoopy, we have implemented functionalities for editing, and animating/simulating coloured qualitative Petri nets (QPN^C), coloured stochastic Petri nets (SPN^C) and coloured continuous Petri nets (CPN^C) [17], [27]. In our implementation, QPN^C is a coloured extension of extended qualitative place/transition net (extended by different types of arcs, e.g. inhibitor arc, read arc, reset arc and equal arc [28]), SPN^C is a coloured extension of biochemically interpreted stochastic Petri nets introduced in [28] and [29], and CPN^C is a coloured extension of continuous Petri nets introduced in [28]. In this paper, the drawing, animation and simulation of coloured Petri net models for PCP are all done by Snoopy.

2 Modelling approach to apply CPN to PCP

We build an abstract model of PCP which only contains the key biological aspects and other relevant information which are essential for the construction of our models. Our study is obviously incomplete, as it does not explicitly identify all relevant genes and molecules, but it provides a useful framework permitting the future undertaking of further research to fulfil the understanding of PCP.

2.1 Abstract model of PCP

In this paper we model the dynamics of the regulatory protein network which controls PCP at two stages of refinement regarding the details of localisation and communication. In the first stage we represent the cell by a highly abstract model, encoded as a (non-coloured) Petri net. The second stage model is more refined and is encoded by a coloured Petri net. Both models describe the cytosol as well as the proximal and distal regions of the cell.

We assume that production of key signalling proteins occurs only in the cytosol and these are degraded constitutively throughout the cell. However, the proteins are distributed asymmetrically within the cell due to an internal transport network. Drosophila wing cells are approximately hexagonal and form a regular honeycomb lattice. The core machinery which controls PCP signalling is uniform across the Drosophila wing. Our model is an abstract description of PCP which includes only the key structure and biological aspects of PCP in order to establish the colour sets principles for each cell, and each compartment within a cell. Therefore, our abstract model is an extremely simplified version of PCP to begin with which only includes essential components and structure and eliminates the duplication of molecular species (places) at the distal and proximal edges of a cell. For example, Fz, Dsh, Pk and Vang exist at both edges of a cell but asymmetrically distribute at a particular edge of the cell, Fz and Dsh at the distal edge while Pk and Vang at the proximal edge. However, they occur only at the particular side of a cell in our abstract model in order to obtain a minimal simplified model which still satisfies the essential need to process the signalling. Thus, in our model, the polarity will be arisen by this asymmetrical distribution of proteins at the distal and proximal edges of each cell together with the intercellular communication. However the power of coloured Petri nets facilitates the construction of a large scale model of PCP in the wing, based on a pattern describing a single cell communicating with its neighbours.

2.2 Simple Petri net model for a single cell

We categorise all reactions involved in each cell into two main types: (1) production and transport of proteins; (2) transformation of proteins (reactions that process the signal). These describe the key biological aspects of PCP and also satisfy our requirement for a simple pattern model which can be used to establish CPN colours for the modelling problem. We firstly sub-divide each cell into four spatial regions: (1) the extracellular space (labelled as communication), where the intercellular complexes form, to the (2) proximal edge (left-hand side of each cell) in order to process intercellular signal between two adjacent neighbouring cells, (3) transport, and (4) distal (right-hand side of each cell). As a result, one cell contains five places (molecular species, A, B, C, D, E), three transitions (reactions, e.g. $r1$) and four spatial regions (e.g. proximal in the blue text box), see Figure 2 for details. In the model, places D_left and E_left indicate that these two molecular species are from the left-hand side neighbouring cell(s).

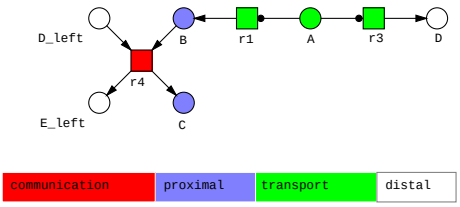


Fig. 2. Petri net model for a single cell: (a) Four spatial regions of a cell: they are labelled as communication, proximal, transport and distal; (b) Places (circle) and transitions (square): they are an abstract representation of the reactions involved in PCP. Production and transport of proteins is represented by places and transitions labelled in green, transformation of proteins is labelled in other colours. It should be noted that the labelled colours here do not provide any information about CPN colorsets , they are only used for demonstration purpose.

2.3 CPN model for pipeline of simple cells

Since PCP exhibits a high replication in terms of reactions and structure, we can simply use the single cell model from the first step as the pattern for the construction of our model of a pipeline of linked simple cells. Thus we create a model which is capable of folding any number of adjacent neighbouring cells using CPN in which a different colour is assigned for each individual cell.

We use the single cell model to start with and then assign a **constant** N to generate N adjacent cells. A **simple colour set** named CS with N colours is created to assign to each place, and a **variable** x with the type CS is assigned for each arc except the one from place D to transition $r4$, which gets the expression $[x > 1] - x$, read as "if x is greater than 1, then it will return the predecessor of x", and meaning that the N cells are linked in a linear pipeline rather than a circuit, see Figure 3 for details.

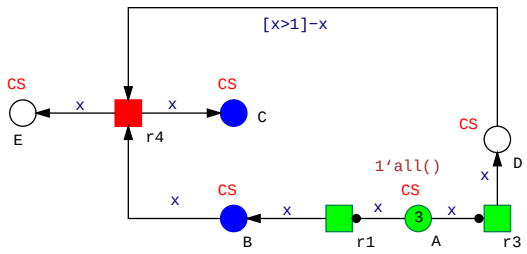


Fig. 3. CPN model for cells linked in a pipeline. The declarations are as follows: colourset $CS = \text{int}$ with $1 - N$, variable x: CS. The arc expression $[x > 1] - x$ indicates that the first cell is not linked to the last, meaning cells are linked in a linear pipeline.

2.4 Refined Petri net model for a single cell

Our refined model exploits the power of CPN to describe repeated structures, and is inspired by Noe et al. [30] who proposed the idea of compartmental kinetic modelling. As described in Section 1.1, the transmission of signalling mainly occurs at the distal and proximal edges of each cell, whereas, the cytosol only involves proteins production and transport. Thus, we need a central compartments to represent the cytosol and several compartments for the distal and proximal edge in each cell. Moreover, we does not consider the communication between the current cell and its north and south neighbouring cells in our model. As a result, we sub-divide each biological cell into seven *virtual compartments* (labelled as number 1, 2, ...7 in blue in Figure 4), three compartments each for the proximal and distal membrane edges, and one compartment for the cytosol, whilst each compartment involves all reactions in the single cell model. Here we aim to establish the framework of applying CPN to PCP signalling – this division of compartments is an initial approach, which will then be further developed in a more sophisticated manner if required. Because Drosophila wing cells form a regular honeycomb lattice there is the need to impose a hierarchical structure over the model, which we express as a regular hexagonal array of cells, each of which comprises seven virtual compartments, see Figure 4.

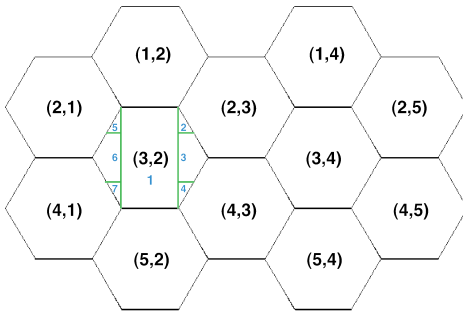


Fig. 4. Compartmentalised Drosophila wing epithelial cell in the context of a fragment of the wing tissue: (1) The coordinate in each cell represents the locality of its corresponding cell in the honeycomb lattice; (2) Each *virtual compartment* in a cell is labelled by number 1 to 7, illustrated by cell $(3, 2)$.

Next we re-construct a Petri net model for a single cell by considering the seven virtual compartments (Figure 5). In this model, each place or transition belongs to a specific compartment, e.g. places D and E are located in three compartments 2, 3, 4 (labelled as $vc2$, $vc3$, $vc4$ in Figure 5) .

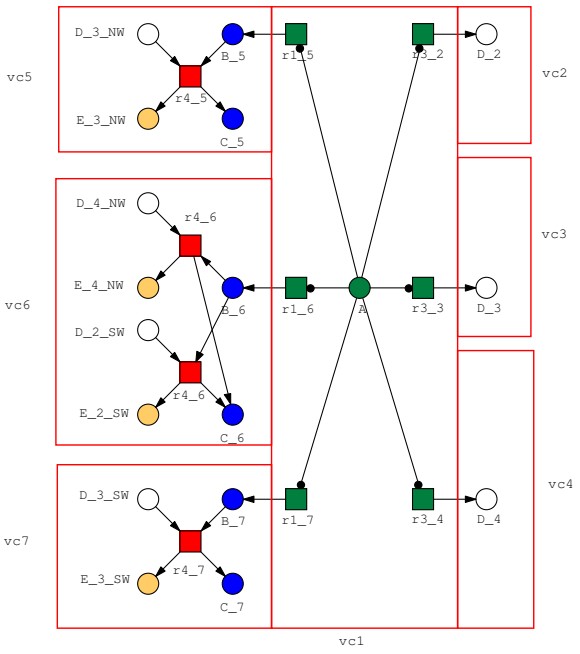


Fig. 5. Refined Petri net model for a single cell with seven compartments (labelled $vc1$, $vc2$,..., $vc7$). Each place or transition belongs to a specific compartment, indicated by a number given as a suffix in place or transition names. NW and SW denote two left neighbours of the current cell.

2.5 CPN model for honeycomb lattice of refined cells

We now describe the construction of a CPN model for PCP with compartment division, following the procedure below.

First, we code cells of PCP as colours of a colour set, i.e. representing the locality of each cell using colours. We have chosen to model a 12-cell fragment of the wing tissue, see Figure 4, as this will give us an adequate size over which to explore the behaviour of our model. From the figure we can see that it is easy to use two-dimensional coordinates e.g. (x, y) to represent the cells in the rectangular honeycomb lattice, which can be defined by the compound colour set *product* in Snoopy. For this, we define two simple colour sets Row and $Column$, denoting the row and column of the lattice respectively, based on which we define a product colour set $CS1$ to represent the coordinates of cells.

Second, we code the virtual compartments as colours. We do not represent them as numbers 1,2,...,7 without considering their localisation within the cell, but use a matrix for compartments, i.e. by using a pair of coordinates (a, b) to denote the location of each compartment in the matrix so that we can clearly

distinguish between them. It should be noted that the middle compartment (cytosol) is represented as three rectangles in the matrix in order to conform with the overall matrix structure, whereas, only one colour set is used for this compartment rather than three colour sets.

Third, we create variables that are used in the guard of transitions and in the expression of arcs connected to transitions. In six virtual compartments for the proximal and distal edges, each arc has been assigned an expression which includes two pairs of coordinate (x, y, a, b), meaning that the arc links the associated place to a particular transition in (a, b) compartment of (x, y) cell. In the middle virtual compartments, the arc expression changes to $(x, y, a + 1, 2)$ and $(x, y, a - 1, 2)$ which indicates the arcs which associate place A to transition $r1$ in proximal (left-hand) compartments are denoted as $a + 1$, while, those link A to the transition $r3$ in distal (right-hand) compartments are denoted as $a - 1$.

Next, we represent the neighbourhood between neighbouring cells. For this, we define two neighbour functions, NW and SW, denoting two left neighbours of the current cell.

Finally, we generate a CPN model for PCP, illustrated in Figure 6. See Table 1 for all declarations.

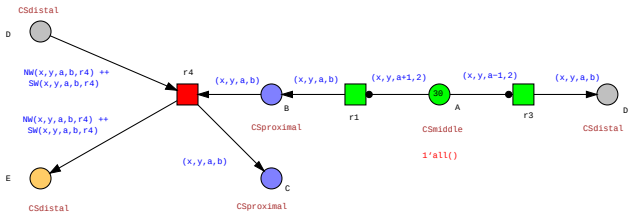


Fig. 6. CPN model describing cells with seven compartments in a 2-D matrix.

Based on what has been obtained from the above models, we will in the future be able to build a more sophisticated model of PCP which includes all detailed reactions according to our current understanding of the biological system. This will facilitate our ability to better understand mechanism of PCP signalling and provide reliable predictions to help guide the design of biological experiments which can help to fill in gaps in our knowledge of the system.

3 Analysis

CPNs enjoy a large variety of analysis techniques, ranging from informal animation or simulation to formal structural analysis or state space analysis. As the models constructed in this paper are still very abstract, we only use animation and stochastic simulation. The analysis reported here was performed on the model which comprises multiple cells and a matrix representing compartments within each cell.

Table 1. Declarations for the coloured Petri net model in Figure 6.

Declaration
colourset Row = int with $1 - M$;
colourset $Column$ = int with $1 - N$;
colourset $ComR$ = int with $1 - R$;
colourset $ComC$ = int with $1 - C$;
colourset $CSr4$ = enum with $c5, c6\_1, c6\_1, c7$;
colourset $CS1$ = product with $Row \times Column$;
colourset $CS2 = CS1$ with $x\%2 = 1\&y\%2 = 0 \vert x\%2 = 0\&y\%2 = 1$;
colourset CS = product with $Row \times Column \times ComR \times ComC$;
colourset $CS4 = CS3$ with $x\%2 = 1\&y\%2 = 0 \vert x\%2 = 0\&y\%2 = 1$;
colourset $CSdistal = CS4$ with $b = 3$;
colourset $CSproximal = CS4$ with $b = 1$;
colourset $CSmiddle = CS4$ with $b = 2$;
variable $x : Row$;
variable $y : Column$;
variable $a : ComR$;
variable $b : ComC$;
variable $r4 : CSr4$;
constant M = int with 5;
constant N = int with 5;
constant C = int with 3;
constant R = int with 3;
function $CSproximal\ NW(Row\ x, Column\ y, ComR\ a, ComC\ b)$;
function $CSproximal\ SW(Row\ x, Column\ y, ComR\ a, ComC\ b)$;

3.1 Animation analysis

We first performed animation analysis (i.e. at the level of the token game) over our CPN model. Our expectation is that protein diffusion is fast. The relevant time-scale in this context is the typical time for diffusion of a membrane protein from one side of a cell to the opposite side which is of the order of 10 minutes. In comparison, the asymmetric pattern of protein localisation arises on a time scale of several hours. This is exactly what our model has shown when we manipulate automatic animation in Snoopy. Thus, it illustrates the reliability of applying CPN to model PCP.

3.2 Stochastic simulation analysis

We use the Gillespie stochastic simulation algorithm (SSA) [31] for the CPN model of PCP in Figure 6. Some of the results that have been produced over an interval of 180 per run are illustrated in Figure 8. The current understanding of the biological system is that the production of the hair is related to the concentration of several species, including actin which is believed to be responsible for the formation of the hair itself. We wish to validate our model by demonstrating that at an abstract level actin is concentrated at the most distal part of a

cell, designated as the future site for prehair formation. In our model, place E represents actin and other key proteins, distributed at the distal edge of the cell (virtual compartments $vc2$, $vc3$ and $vc4$) during signalling. Figure 7 shows how E changes over time in the three distal compartments ($vc2$, $vc3$ and $vc4$) of cell $(3,4)$ and Figure 8 displays the final concentration of the coloured place E in virtual compartments $vc2$, $vc3$ and $vc4$ for each of the 12 cells (refer to Figure 4 for mappings). The results clearly show that the major accumulation of actin occurs in virtual compartment $vc3$ for each of the cells except for cells (2,5) and (4,5) which do not have distal neighbours and thus lack inter-cellular communication in that direction. The accumulation of actin in $vc3$ corresponds exactly to the location of the prehair formation at the most distal vertex of each cell, see Figure 1, and we find that it is highest in $vc3$ for cells (3,2) and (3,4) which have the maximum number of neighbouring cells (6 each) in the honeycomb lattice.

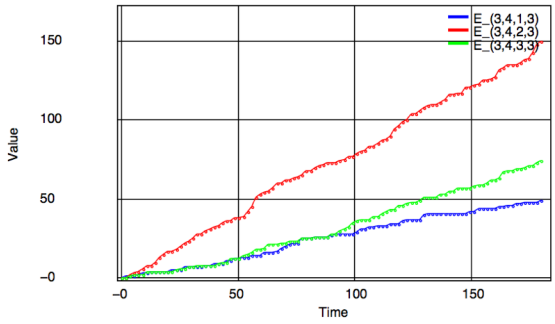


Fig. 7. Stochastic simulation result: time course plots for the value of E within cell $(3,4)$ in the three virtual compartments $vc2$ (3,4,1,3), $vc3$ (3,4,2,3) and $vc4$ (3,4,3,3). Refer to Figure 4 for mappings.

		0			0			
	72	(1,2) **97**		69	(1,4) **77**		0	
(2,1)	**135**	49	(2,3)	**148**	45	(2,5)	0	
	46	58		45	50		0	
		(3,2) **168**			(3,4) **170**			
	46	48		49	39		0	
(4,1)	**167**	42	(4,3)	**165**	43	(4,5)	0	
	76	(5,2) **88**		68	(5,4) **84**		0	
		0			0			

Fig. 8. Stochastic simulation result: final values for E in virtual compartments $vc2$, $vc3$ and $vc4$ for each of the 12 cells which are labelled by their identification tuple, refer to Figure 4 for mappings.

4 Conclusion

In this paper, we have presented our current work applying CPN techniques to construct a PCP model in order to explore the mechanisms that drive PCP. Our aim has been to provide a proof of principle for the use of CPN to model a multi-cellular system with a hierarchical structure while keeping the model mathematically tractable.

The model we have developed has allowed us to generate behaviours as a first step to explaining the complex behaviours observed in the biological system and to explore the implications of variations in the model. Our analysis confirms that the behaviour of the model correctly shows the major accumulation of actin occurring in the most distal part of the cell, corresponding to the location of the prehair formation in wing cells of Drosophila.

However, the ability of the current model to make predictions and provide an accurate picture of PCP signalling is limited by its lack of biological detail. In ongoing work we are refining this abstract model into a more detailed model, which includes exploring alternative ways in which to model the cellular machinery of PCP signalling. With this refined model we will be able not only to perform simulations of PCP signalling in wild-type cells but also on patches of mutant cells in a wild-type background. Our long term goal is to facilitate a better understanding of the mechanisms that drive PCP, and to make predictions about the behaviour of the system when it is perturbed by the mutation of specific genetic components.

References

1. Heath, A. P., Kavraki, L. E.: Computational challenges in systems biology. Computer Science Review. 3 (1), 1-17 (2009)
2. Reddy, V. N., Mavrovouniotis, M. L., Liebman, M. N.: Petri Net Representations in Metabolic Pathways. Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology. ISBN: 0-929280-47-4, vol. 328–336, pp. 9. AAAI Press (1993)
3. Gilbert, D., Heiner, M.: From Petri nets to differential equations - an integrative approach for biochemical network analysis. Lecture Notes in Computer Science. 4024,181-20 (2006)
4. Baldan, P., Cocco, N., Marin, A., Simeoni, M.: Petri nets for modelling metabolic pathways: a survey. Natural Computing. 9 (4), 955-989 (2010)
5. Marwan, W., Wagler, A., Weismantel, R.: Petri nets as a framework for the reconstruction and analysis of signal transduction pathways and regulatory networks. Natural Computing. Vol. 9 (2009)
6. Genrich, H., Kffner, R., Voss, K.: Executable Petri net models for the analysis of metabolic pathways. International Journal on Software Tools for Technology Transfer. 3 (4), 394-404 (2001)
7. Lee, D., Zimmer, R., Lee, S., Park, S.: Colored Petri net modeling and simulation of signal transduction pathways. Metabolic Engineering. 8, 112-122 (2006)
8. Peleg, M., Gabashvili, I. S., Altman, R. B.: Qualitative models of molecular function: linking genetic polymorphisms of tRNA to their functional sequelae. Proceedings of the IEEE. 90 (12), 1875-1886 (2002)

9. Tubner, C., Mathiak, B., Kupfer, A., Fleischer, N., Eckstein, S.: Modelling and simulation of the TLR4 pathway with coloured Petri nets. Proceedings of the 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 2009-2012. IEEE, New York (2006)
10. Heiner, M., Koch, I., Voss, K.: Analysis and simulation of steady states in metabolic pathways with Petri nets. In: K. Jensen (ed) Proceedings of the Third Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, vol. 29-31, pp. 15-34. University of Aarhus, Aarhus (2001)
11. Voss, K., Heiner, M., Koch, I.: Steady state analysis of metabolic pathways using Petri nets. In Silico Biology. 3, 0031 (2003)
12. Runge, T.: Application of coloured Petri nets in systems biology. In: AarhusK Jensen (ed) Proceedings of the fifth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, pp. 77-95. University of Aarhus, Aarhus (2004)
13. Bahi-Jaber, N., Pontier, D.: Modeling transmission of directly transmitted infectious diseases using colored stochastic petri nets. Mathematical Biosciences. 185, 1-13 (2003)
14. Christensen, S., Jrgensen, J. B., Kristensen, L. M.: Design/CPN - A Computer Tool for Coloured Petri Nets. Proceedings of the Third International Workshop on Tools and Algorithms for Construction and Analysis of Systems. LNCS 1217, pp. 209-223. Springer-Verlag London (1997)
15. Jensen, K., Kristensen, L. M., Wells, L. M.:. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer. 9 (3/4), 213-254 (2007)
16. Rohr, C., Marwan, W., Heiner, M.: Snoopy - a unifying Petri net framework to investigate biomolecular networks. Bioinformatics. 26(7), 974-975 (2010)
17. Liu, F., Heiner, M.: Colored Petri nets to model and simulate biological systems. Int. Workshop on Biological Processes and Petri Nets (BioPPN), satellite event of Petri Nets 2010. Braga, Portugal (2010)
18. Simons, M., Mlodzik, M.: Planar cell polarity signaling: From fly development to human disease. Annu. Rev. Genet. 42, 517-540 (2008)
19. Wong, L.L., Adler, P.N.: Tissue polarity genes of drosophila regulate the subcellular location for prehair initiation in pupal wing cells. J. Cell Biol. 123, 209-220 (1993)
20. Strutt, D.I.: Asymmetric localization of frizzled and the establishment of cell polarity in the drosophila wing. Mol. Cell. 7, 367-375 (2002)
21. Amonlirdviman, K., Khare, N.A., Tree, D.R.P., Chen, W.-S., Axelrod, J.D., Tomlin, C.J.: Mathematical modeling of planar cell polarity to understand domineering nonautonomy. Science. 307, 423-426 (2005)
22. Raffard, R.L., Amonlirdviman, K., Axelrod, J.D., Tomlin, C.J.: An adjoint-based parameter identification algorithm applied to planar cell polarity signaling. IEEE Trans. Automat. Contr. 53, 109-121 (2008)
23. Le Garrec, J.F., Lopez, P., Kerszberg, M.: Establishment and maintenance of planar epithelial cell polarity by asymmetric cadherin bridges: A computer model. Dev. Dyn. 235, 235-246 (2006)
24. Le Garrec, J.F., Kerszberg, M.: Modeling polarity buildup and cell fate decision in the fly eye: Insight into the connection between the PCP and Notch pathways. Dev. Genes Evol. 218, 413-426 (2008)
25. Jensen, K.: Coloured Petri Nets and the Invariant-Method. Theor. Comput. 14, 317-336 (1981)
26. Jensen, K., Kristensen, L. M.: Coloured Petri nets. Springer (2009)

27. Liu, F., Heiner, M.: Computation of Enabled Transition Instances for Colored Petri Nets. Proc. 17th German Workshop on Algorithms and Tools for Petri Nets (AWPN 2010), CEUR-WS.org. CEUR Workshop Proceedings, vol. 643, pp. 51–65, http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-643/paper05.pdf (2010)
28. Gilbert, D., Heiner, M., Lehrack, S.: A unifying framework for modelling and analysing biochemical pathways using Petri nets. LNCS/LNBI. 4695, 200-216 (2007)
29. Heiner, M., Lehrack, S., Gilbert, D., Marwan, W.: Extended Stochastic Petri Nets for Model-based Design of Wetlab Experiments. Transaction on Computational Systems Biology XI. LNCS/LNBI. 5750, 138-163 (2009)
30. Noe, D.A., Delenick, J.C.: Quantitative analysis of membrane and secretory protein processing and intracellular transport. J. Cell Sci. 92, 449-459 (1989)
31. Gillespie, D T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry. 81 (25), 2340-2361 (1977)