Chapter 1
Infobiotics Workbench - A P Systems based Tool for Systems and Synthetic Biology

Jonathan Blakes[1], Jamie Twycross[1], Savas Konur[2],
Francisco Jose Romero-Campero[3], Natalio Krasnogor[1] and Marian Gheorghe[2]

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

1.1 Introduction

Membrane computing is a growing area of research in computer science and, more specifically, natural computation. Membrane computing assumes that the processes taking place in the compartments of a living cell can be interpreted as computations. The devices of this model are called P systems. A P system consists of a cell-like membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules. Because a set of rules is a mathematical entity, it can be analysed with formal rigour to discover the relationships between rules and their subjects, potential sequences of events, and the reachability of certain states.

The Infobiotics Workbench is an integrated *stochastic P systems* based platform for computer-aided modelling, design and analysis of large-scale biological systems which consists of three key components: (a) a *simulator for a modelling language*

[1] ICOS Research Group, School of Computer Science, University of Nottingham, UK
e-mail:{jonathan.blakes, jamie.twycross}@nottingham.ac.uk
natalio.krasnogor@nottingham.ac.uk
[2] Department of Computer Science, University of Sheffield, UK
e-mail:{s.konur, m.gheorghe}@sheffield.ac.uk
[3] Department of Computer Science and Artificial Intelligence, University of Seville, Spain
e-mail: fran@us.es

- *discussed in Section 1.4.2*; (b) a *model checking* module - see Section 1.4.3; and (c) a *model structure and parameter optimisation* engine - details in Section 1.4.4. The availability of deterministic and multi-compartment stochastic simulation of population models enables comparisons between macroscopic and mesoscopic interpretations of molecular interaction networks and investigation of temporo-spatial phenomena in multicellular systems. Model checking can be used to increase confidence in simulated observations by quantifying the probability of reaching definable states for all possible trajectories [76]. The optimisation component of the Workbench enables designs of synthetic circuits matching a set of desired temporal dynamics (specified as time series of molecular species quantities) to be automatically composed from modules of abstract networks motifs and/or completely specified bioparts (with corresponding DNA sequences) drawn from libraries of reusable model components.

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

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

1.2 Overview

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

1.2.1 Mathematical continuous models

The vast majority of models used in systems biology have, until recently, been mathematical, based on systems of coupled ordinary differential equations (ODEs). In an

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

1.2.2 Stochastic discrete models

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

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

The CME represents a continous-time Markov chain which can capture the noise (stochasticity) in the system. Unfortunately the CME is actually a system of as many coupled ordinary differential equations as there are combinations of molecules that can exist in the system, and can only be solved analytically for a very few simple systems [59]. Fortunately a more tractable approach exists. Instead of solving the CME we can construct numerical realisations of the system's state over time, that is, generate trajectories of the system using a kinetic Monte Carlo algorithm, Gillespie's *stochastic simulation algorithm* (SSA) [56], in *exact* compliance with the CME.

Gillespie initially produced two SSAs that simulate every reaction in the system: the First Reaction Method [55] and the simpler but equivalent Direct Method [56]; and subsequently showed these to be a rigorous derivation of the CME [57]. More efficient exact SSAs have been introduced since, including the dominant Next Reaction Method (NRM) [52] which scales logarithmically with the number of reactions, the Next Subvolume Method [39] as a variation on NRM for discrete-space intracellular models, the Partial-propensity Direct Method [108] scaling at most linearly with the number of species (often far fewer than reactions), and the Composition-Rejection SSA [120] offering constant-time performance for 10^5 or greater reactions. *Approximate* methods, that simulate batches of fast non-critical reactions, include τ-leaping (established in [58] and optimised in [21]) and the slow-scale SSA [22]. These offer accelerated performance for stiff systems, with an acceptable and tunable loss of accuracy, and enable larger models to be simulated in reasonable time.

1.2.3 Executable modeling formalisms

The formalisation of biological systems using alternatives to mathematical equations has recently received much interest as a deeper mechanistic understanding of biological systems is sought through modelling. Formalisms where molecular populations and interactions are modelled as discrete entities and events have come to be known collectively as *Executable Biology*. *Executable biology* [44, 43], or *algorithmic systems biology* [103], propose the application of established computational formalisms from other domains, and domain specific languages for the formalisation and implementation of biological models. Below we review a selection of these alternative representations, their capabilities and implementations.

The Systems Biology Markup Language (SBML) [70] is an XML dialect used to store and exchange models of biological systems between different tools. SBML files store information about model compartments, species and reactions, as well as events, units, etc. that are relevant to some models and approaches but not others. Tools for the visual specification of models in SBML, e.g. *CellDesigner* [48], *e-cell* [128], *VCell* [88] and *COPASI* [69], enable the visual creation of models from a collection of symbols for various types of molecular and interactions.

Cellular automata were studied in the early 1950s as a possible model for biological systems ([127], p48). This formalism, inspired from cellular biology, has been extensively used in modelling a broad spectrum of biological systems, amongst them pattern formation (morphogenesys) [31], ecology and population biology, immunology, oscillations, diffusion processes, fibroblast aggregation, ant trails and others (for more details see the overview paper [40]). In the paper coining the term algorithmic systems biology, cellular automata are mentioned amongst the models employing explicitly computational aspects [104].

Cellular automata have been connected to membrane systems for different modelling reasons. In [26] it is studied the behaviour of HIV infection by comparing a cellular automaton model and a conform-P system model with respect to the robustness related to various initial conditions and parameters. The possibility of converting a cellular automaton into a generalised P systems has been also investigated [90].

Boolean networks [73] are one of the oldest examples of executable biological modelling formalisms. They represent the interactions of genes as a directed graph. Each node of a Boolean network can represent a gene that is either active, or inactive. Edges between nodes contribute either positively (activation) or negatively (inactivation) to the node at which they are directed (providing the node from which the edge extends is active), modelling hierarchies of genetic regulations. Boolean networks are deterministic given their starting configuration for which there are 2^n possible system-wide states where n is the number of nodes.

Boolean networks are qualitative in terms of quantities and time. With only topological data and binary relationships required to build a model, Boolean networks can usually be constructed when data is scarce, and are therefore often chosen as a modelling formalism for their amenability to analysis rather than realism [42].

Similarly qualitative but more fine-grained are **Statecharts**, a method devised for the engineering of complex reactive systems. Statecharts have been used to successfully model the interactions of two signalling pathways, specifying the fates of the six vulval precursor cells, which provide a mechanism for pattern formation during the C.elegans development [45].

Petri nets are formalisms that model systems with concurrent behaviour and are particularly suited to modelling discrete asynchronous distributed systems. Petri nets were initially applied to biological pathways [110, 109] for semi-quantitative analysis in terms of discrete number of objects and uniform time intervals. A bibliography [126] of Petri nets applications in biomolecular modelling, simulation and analysis summarises developments up to 2002. More recent contributions include the ubiquitously studied ERK signal transduction pathway [54], receptor signalling and kinase cascades, cell-cycle regulation and wound healing [53], and synthetic biology [66].

A quantitative notion of time is introduced by *stochastic Petri nets* [89, 123], where each transition has an associated rate from which a period of time is calculated upon firing and added to the global clock, typically using a stochastic simulation algorithm. *Coloured Petri nets* [72] can provide a novel way of dealing with the combinatorial explosion of states, where differently coloured tokens can represent molecules of the place's species with various modifications, or alternatively molecules in different cells without extrapolating the Petri net [50].

There are numerous tools deployed to create and analyse Petri nets. We refer the reader to [98] for the database of the available tools.

Process algebras (or process calculi) are a diverse family of related formalisms that describe distributed concurrent processes, such as the objects inside a computer program or a collection of programs, interacting. π-calculus [87], for concurrent mobile processes, is an accepted model for interacting systems with communication topologies that evolve dynamically [93]. For biological models, process algebras consider molecules with binding sites as processes with communication channels.

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

Stochastic π-calculus (initially proposed as Sπ [102]) enables fully quantitative simulations by associating a rate constant with each channel. BioSPI [106], the first stochastic π-calculus simulator [111], could simulate systems with hundreds of processes in the order of seconds [105]. The current leading implementation of a stochastic π-calculus simulator is SPiM [99]. A more intuitive understanding of π-calculus is made possible by a graphical representation [100] that visualises the state-space of each process as a graph and has been incorporated into SPiM. In [100] a graphical execution model was defined and proved equivalent to Sπ.

Performance Evaluation Process Algebra (PEPA) is an alternative stochastic process algebra that has been applied to modelling signalling pathways [17, 16, 18, 15] and synthetic biology designs [51]. PEPA can be used for reagent-centric and pathway-centric modelling [17]. *Bio-PEPA* [25] is a biologically-oriented modification of PEPA incorporating stoichiometry and the use of kinetic laws in rate functions.

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

We refer the reader to [62] for an extensive review on the application of process algebras to biological modelling up to 2006. Other notable works include GEC [94] and LCS [95].

Membrane computing [92] is a branch of natural computing that emphasises the compartmentalised nature of biological systems and its power in computation. The central objects are P systems, that consist of a membrane structure, the regions of which contain rewriting rules operating on multisets of objects [114]. The P system *evolves* by the repeated application of rules, mimicking chemical reactions and transportation across membranes, and halts when no more rules can be applied.

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

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

  - *Metabolic P systems* (MP systems) have diverged considerably from the non-deterministic, compartmentation-based notion of P systems, being coarse-grained models of the fluxes between molecular populations within a single membrane computed by means of the metabolic algorithm (MA) - its equational formulation is in [80]. A methodology for inferring and validating the model has been elaborated [83]. An overall presentation of these systems is available in [81] and a comprehensive description in [82]. MP systems are supported by the MetaPlab [86] software, previously Psim [11].
  - *Non-deterministic P systems* are used in a context where the rules are selected according to a waiting time algorithm involving a mass action law principle [71]; this model is successfully utilised to analyse the behaviour of different biochemical signalling networks. Another special class of P systems, called *conformon-P systems*, deals with systems having rewriting and communication rules using together with multisets, some numerical values that help controlling the computation. These models have been used to study how some diseases spread [26].

- *Probabilistic (stochastic) P systems* include several classes of P systems:

  - *Stochastic P systems (SP systems)* [117] directly apply stochastic rate constants and Gillespie's stochastic simulation algorithms to P systems, with boundary rules that make the specification of molecule transport between enclosed and enclosing members simple and intuitive. These are discussed in much greater detail in Section 1.3.
  - *Dynamical Probabilistic P systems* (DPP) [97] use standard P systems with a novel rule application method to model biological phenomena in a discrete and stochastic way (motivated by the investigation of maximal parallelism in nature). In a procedure not unlike propensity calculation in the Gillespie algorithm DPP rules are dynamically assigned a probability that is the product of the possible combinations of reactant objects and an associated rate. A tau leaping variant of it is also provided [24] which is packaged in the BioSimWare software platform [7].
  - *Probabilistic Dynamics Population* P systems represent a class of P systems meant to provide an accurate model of multi-environmental systems; it has applications to ecosystems, where the methodology consists of a modular specification including probabilistic rules [28] describing transformations within compartments as well as communications between compartments and cooperations involving different parts of the environment. This approach is included in the P-Lingua framework [118] and has a number of implementations, including one that uses GPU hardware [84]. An integrated software environment, called MeCoSim, is supporting the modelling language with an editor and different visualisation options [96].
  - *Probabilistic P systems* with peripheral proteins focuses on trans-membrane operations where a Gillespie algorithm is used for describing the system behaviour; a specification language is integrated into the simulation environment Cyto-sim [23].

- *Extension of P systems with string objects* for modelling protein binding domains with ligands have been considered for specifying oscillatory phenomena; a software environment, called SRSim, which incorporates spatial rules and a strong visualisation engine is available [68].

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

A general-purpose class of computational tools has been introduced for tackling the challenge of a combinatorial explosion in the number of interactions that arises when many species with coincidental modifications, conformations or states need to be represented explicitly. Some of the most prominent rule-based systems that deal with these issues are NFsim [121], BioNetGen [41], Kappa [27] and little b [79]. While each of these approaches can model some aspects regarding pathways and their molecular components, none of the approaches can fully capture "quantitative dynamics, interactions among molecular entities and structural organisation of cells" [114].

1.3 Lattice Population P systems

Many multicellular biological systems have a spatial component where molecule exchange between *adjacent* cells determines the overall phenotypes. However, this structure cannot be captured by stochastic P systems, which have only a hierarchical membrane structure of compartments within other compartments or a simple population of such entities. Therefore, stochastic P systems need to be augmented with an additional level of organisation, a 2-dimensional geometric lattice on which a population of P systems can be placed and over which molecules can be translocated. Rules that move objects from one P system to another on the lattice are associated a vector that describes where to put that molecules. We call this extension of stochastic P systems *Lattice Population P systems* (*LPP systems* for short) and, in the tradition of P systems, proceed with their formal definition (published in [117]).

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

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

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

$$SP = (O, L, \mu, M_1, \ldots, M_n, R_1, \ldots, R_n) \tag{1.1}$$

where:

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

$$r_i^{l_k} \ : \ o_1 \ [\ o_2 \ ]_l \xrightarrow{c_i^{l_k}} o_1' \ [\ o_2' \ ]_l \tag{1.2}$$

where o_1, o_2 and o_1', o_2' are multisets of objects (possibly empty), over O, representing the molecular species consumed and produced in the corresponding molecular interaction. The square brackets and the label l describe the compartment involved in the interaction. An application of a rule of this form changes the content of the membrane with label l by replacing the multisite o_2 with o_2' and the content of the membrane outside by replacing the objects o_1 with o_1'. The stochastic constant $c_i^{l_k}$ is used to compute the propensity of the rule by multiplying it by the number of available reactants in the membrane, where the same object is not counted twice for homogenous bimolecular reactions [6]. The propensity associated with each rule is used to compute the probability and time needed to apply it (according to the stochastic semantics of Gillespie's theory of chemical kinetics [55]).

Definition 1 provides the formalism needed for the specification of an individual cell with its structure given by μ and the outer membrane called the *skin membrane*. To specify the possible spatial distribution of cells assembled into colonies and tissues we define an array of regularly distributed points according to a *finite point lattice* or *grid* [78] capable of describing the spatial geometries (see Fig. 1.1).

This model looks very similar to a cellular automaton although in lattice population P systems we have considered that each cell of the grid has a cell-like stochastic

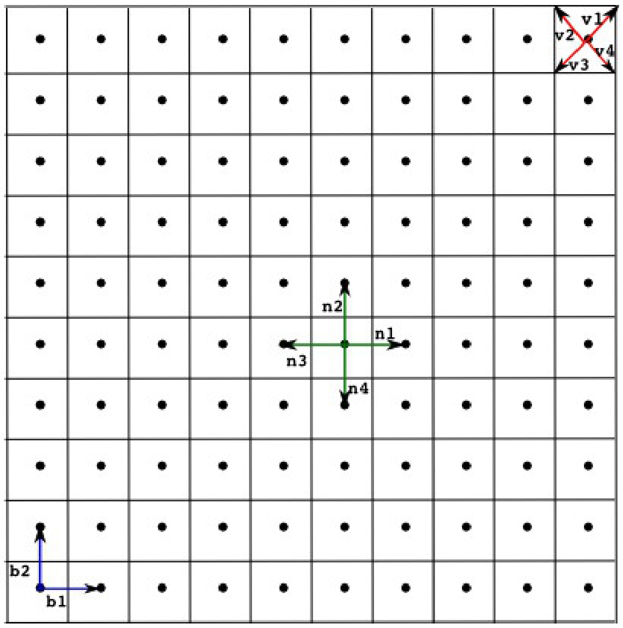


Fig. 1.1: A square lattice.

membrane system inside and this type of grid has been chosen to illustrate a specific geometry we have considered so far. Our model is more general than a cellular automaton and in the future some more complex geometries describing 3D complex structures will be introduced.

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

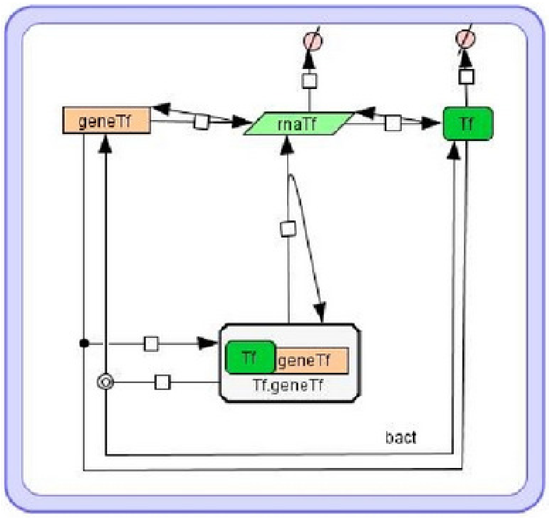
$$Lat = (B, o, (\alpha_1^{min}, \alpha_1^{max}, \ldots, \alpha_n^{min}, \alpha_n^{max})) \tag{1.3}$$

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

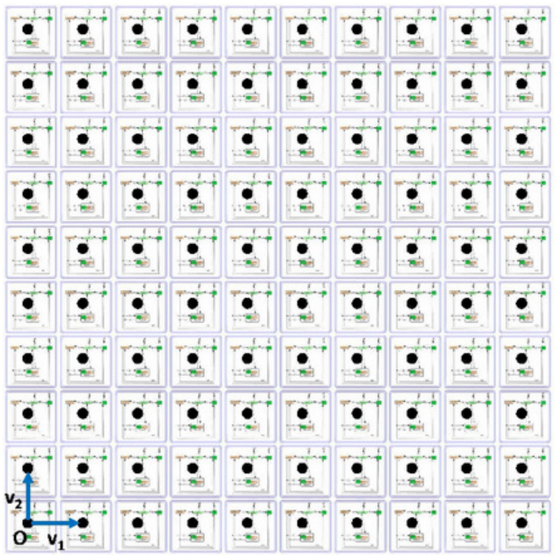
$$P(Lat) = \{o + \sum_{i=1}^{n} \alpha_i v_i \ : \ \forall i = 1, \ldots, n \ (\alpha_i \in \mathbb{Z} \wedge \alpha_i^{min} \leq \alpha_i \leq \alpha_i^{max})\} \tag{1.4}$$

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

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



(a)



(b)

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

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

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

where

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

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

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

Molecular reaction networks can, to a certain degree, be decomposed into modules acting as discrete entities carrying out particular tasks [65]. It has been shown that there exist specific modules termed *motifs* that appear recurrently in transcriptional networks performing specific functions like response acceleration and noise filtering [1]. Modularisation is also a central technique used in the engineering of synthetic cellular systems by combining well-characterised and standardised cellular models [19] as exemplified in the MIT BioBricks project [119].

Definition 4 gives the definition of a *P system module* that we use [115] to decompose large sets of rules into more meaningful and reusable subsets. Other similar concepts of modularity in P systems for various other classes of P systems. Modules of a conformon-P systems are discussed in [47]. In [32] P modules are introduced with the aim of facilitating a modular decomposition of complex P systems, whereas in [67] it is defined as a functional unit fulfilling some elementary computational tasks. In the context of generalised communicating P systems [125] it is introduced a concept of a module as a network of cells. Subsequently we introduce the concept of a module for stochastic P systems with the aim of capturing some high level behaviour which can be characterised by some specific parameters and which outlines some generic names that are instantiated with specific values in various contexts.

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

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

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

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

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

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

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

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

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

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

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

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

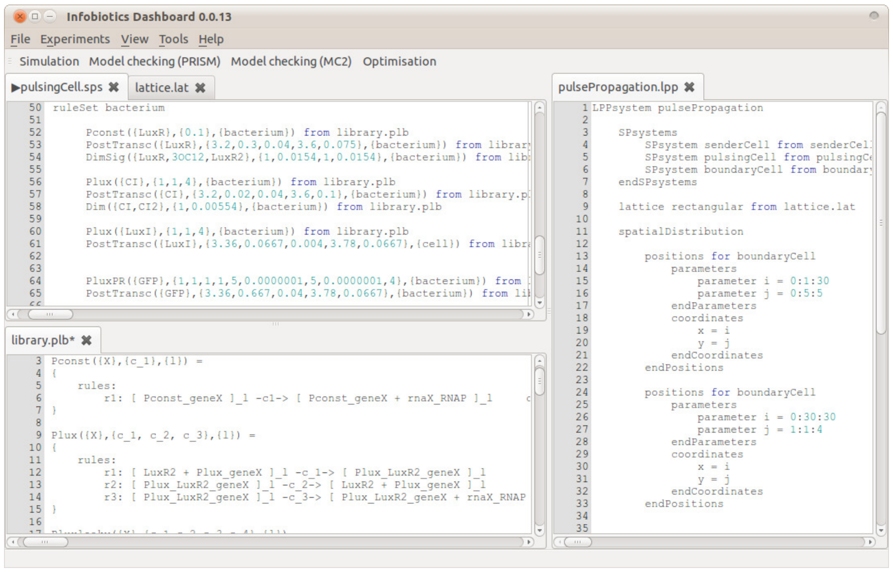
1.4 Infobiotics Workbench

The Infobiotics Workbench (IBW)[1] is an integrated software suite of tools to perform *in silico* experiments for LPP models in Systems and Synthetic Biology [14]. Models are simulated either using stochastic simulation or deterministic numerical integration using MCSS, an application for simulating multi-compartment stochastic P system models, and visualised in time and space with the *Infobiotics Dashboard*.

[1] `http://www.infobiotics.org`

Model structure and parameters can be optimised with evolutionary algorithms using POPTIMIZER, and properties of a model's temporo-spatial behaviour calculated using probabilistic or simulative model checking with PMODELCHECKER.

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


Infobiotics Dashboard 0.0.13
File Experiments View Tools Help
Simulation Model checking (PRISM) Model checking (MC2) Optimisation
▶pulsingCell.sps
lattice.lat
library.plb*
pulsePropagation.lpp

```
ruleSet bacterium

      Pconst({LuxR},{0.1},{bacterium}) from library.plb
      PostTransc({LuxR},{3.2,0.3,0.04,3.6,0.075},{bacterium}) from librar
      DimSig({LuxR,3OC12,LuxR2},{1,0.0154,1,0.0154},{bacterium}) from lib

      Plux({CI},{1,1,4},{bacterium}) from library.plb
      PostTransc({CI},{3.2,0.02,0.04,3.6,0.1},{bacterium}) from library.p
      Dim({CI,CI2},{1,0.00554},{bacterium}) from library.plb

      Plux({LuxI},{1,1,4},{bacterium}) from library.plb
      PostTransc({LuxI},{3.36,0.0667,0.004,3.78,0.0667},{cell}) from libr


      PluxPR({GFP},{1,1,1,1,5,0.0000001,5,0.0000001,4},{bacterium}) from
      PostTransc({GFP},{3.36,0.667,0.04,3.78,0.0667},{bacterium}) from li
```

```
Pconst({X},{c_1},{l}) =
{
    rules:
        r1: [ Pconst_geneX ]_l -c1-> [ Pconst_geneX + rnaX_RNAP ]_l     c
}

Plux({X},{c_1, c_2, c_3},{l}) =
{
    rules:
        r1: [ LuxR2 + Plux_geneX ]_l -c_1-> [ Plux_LuxR2_geneX ]_l
        r2: [ Plux_LuxR2_geneX ]_l -c_2-> [ LuxR2 + Plux_geneX ]_l
        r3: [ Plux_LuxR2_geneX ]_l -c_3-> [ Plux_LuxR2_geneX + rnaX_RNAP
}

```

```
LPPsystem pulsePropagation

    SPsystems
        SPsystem senderCell from senderCel
        SPsystem pulsingCell from pulsingC
        SPsystem boundaryCell from boundar
    endSPsystems

    lattice rectangular from lattice.lat

    spatialDistribution

        positions for boundaryCell
            parameters
                parameter i = 0:1:30
                parameter j = 0:5:5
            endParameters
            coordinates
                x = i
                y = j
            endCoordinates
        endPositions

        positions for boundaryCell
            parameters
                parameter i = 0:30:30
                parameter j = 1:1:4
            endParameters
            coordinates
                x = i
                y = j
            endCoordinates
        endPositions


```

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

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

1.4.1 Modelling in LPP Systems

For LPP system models to be specified and manipulated by computers it is necessary that they have a machine-readable equivalent. LPP system XML is a set of machine-readable data formats which closely mirrors our formal definitions. It allows us to define, in a single file or multiple files, modules of stochastic P system rules, P

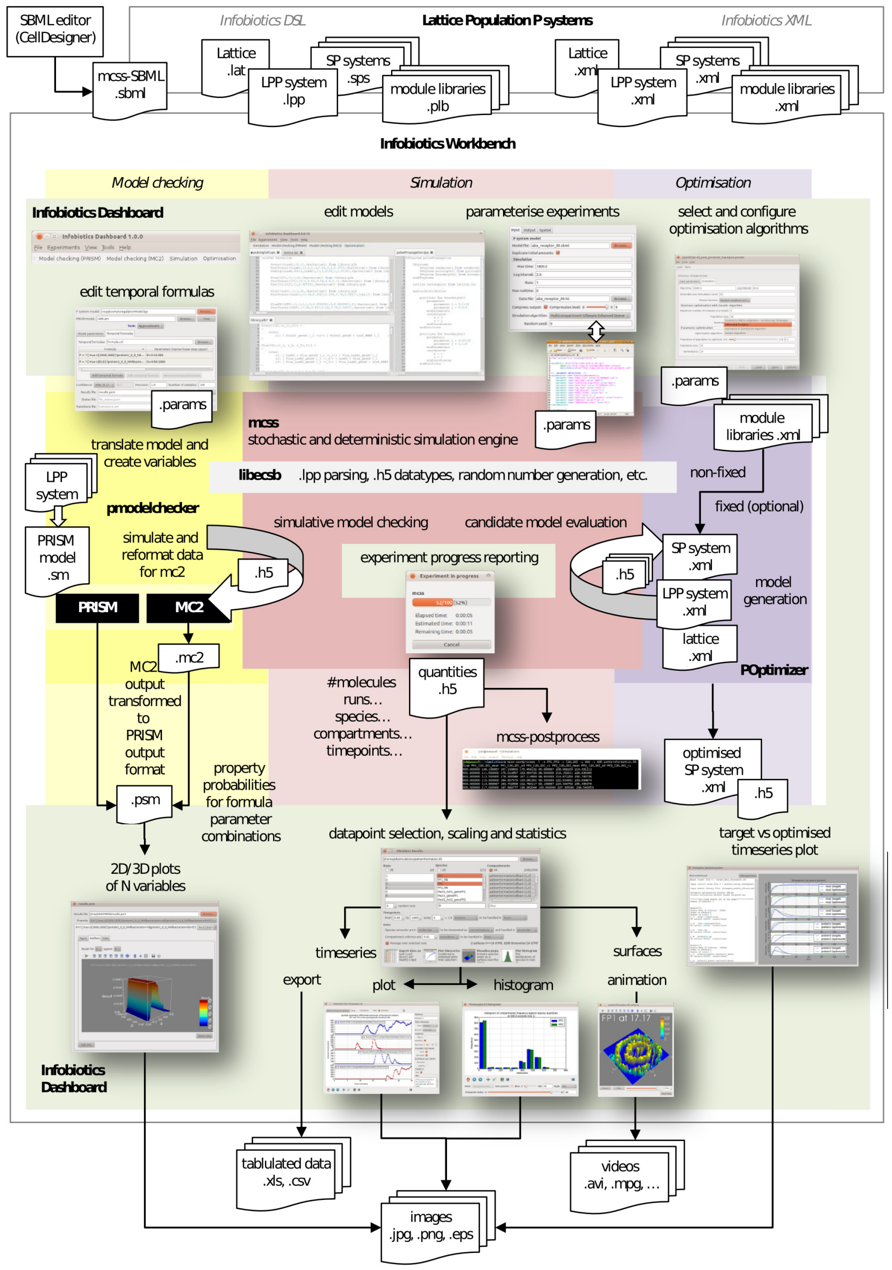


Fig. 1.4: Flow of information through the components of the Infobiotics Workbench. Data is passed between components as files. Parameter files (`.params`), referencing model files (`.sbml`, `.lpp` or `.xml`), are produced by the Infobiotics Dashboard and supplied to the experiment executables for simulation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

systems with initial multisets and instantiations of modules of rules, a geometric lattice and distribution of P systems over the lattice, which together constitute an LPP system model.

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

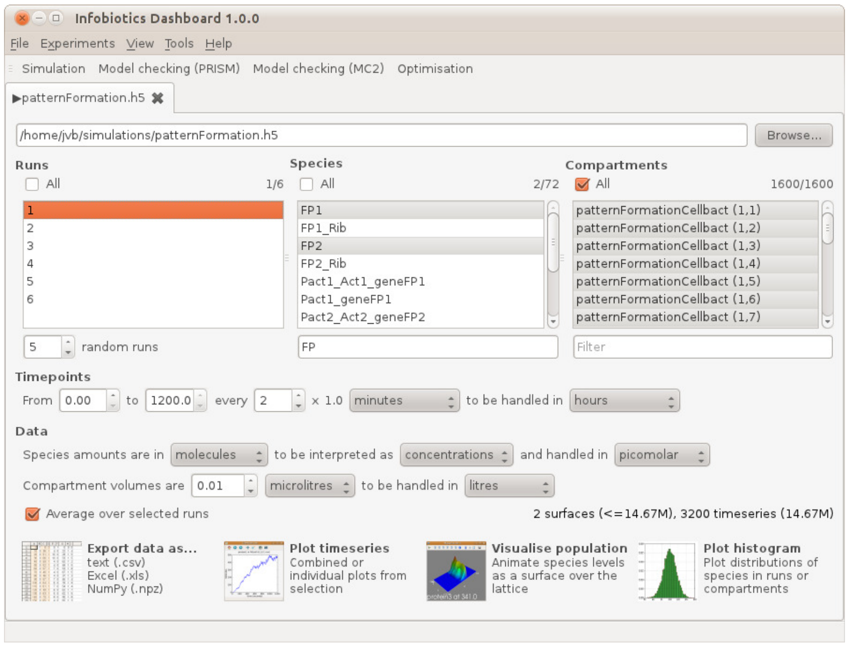
The LPP formalism enables three types of modelling component reuse:

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

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

1.4.2 Simulation

Simulation recreates the dynamics of a system as described by a model. Quantitative simulations enables measurements of model features changing in time which can be compared with observations of the real system for validation and predictive purposes. The Infobiotics Workbench simulator, MCSS, offers a choice of two types of quantitative simulations: deterministic numerical approximation with standard solvers, and execution of the model with stochastic simulation algorithms. In addition to providing a baseline implementation of the canonical Gillespie Direct Method, MCSS implements an optimised multi-compartmental SSA with queue [116] that takes advantage of the compartmentalised nature of LPP system models by storing the next reaction to fire for each compartment in the heap and only recalculating the propensities of the reactions in the compartments where a reaction occurs, both compartments involved in a species translocation. This greatly improves performance, decreasing the simulation time of models with tens of thousands of compartments and hundreds of reactions and species per compartment.


Infobiotics Dashboard 1.0.0
File Experiments View Tools Help
Simulation Model checking (PRISM) Model checking (MC2) Optimisation
patternFormation.h5
Browse...
Export data as... text (.csv) Excel (.xls) NumPy (.npz)
Plot timeseries Combined or individual plots from selection
Visualise population Animate species levels as a surface over the lattice
Plot histogram Plot distributions of species in runs or compartments

/home/jvb/simulations/patternFormation.h5

Runs: All 1/6 — 1, 2, 3, 4, 5, 6 — 5 random runs

Species: All 2/72 — FP1, FP1_Rib, FP2, FP2_Rib, Pact1_Act1_geneFP1, Pact1_geneFP1, Pact2_Act2_geneFP2 — FP

Compartments: All 1600/1600 — patternFormationCellbact (1,1), patternFormationCellbact (1,2), patternFormationCellbact (1,3), patternFormationCellbact (1,4), patternFormationCellbact (1,5), patternFormationCellbact (1,6), patternFormationCellbact (1,7) — Filter

Timepoints: From 0.00 to 1200.0 every 2 x 1.0 minutes to be handled in hours

Data: Species amounts are in molecules to be interpreted as concentrations and handled in picomolar

Compartment volumes are 0.01 microlitres to be handled in litres

Average over selected runs — 2 surfaces (<=14.67M), 3200 timeseries (14.67M)

Fig. 1.5: The simulation results interface.

In order to perform deterministic simulations MCSS derives a set of ordinary different equations from the stochastic rules of the entire LPP system: each pool of identical objects in different compartments is treated as a separate continuous variable whose rate of change is determined by mass-action kinetics involving only the variables corresponding to reactants and products of those rules affecting the pool. A solution of the resultant equations is obtained using algorithms provided by the GNU Scientific Library (GSL) [49], including explicit 4th order Runge-Kutta and implicit ODE solvers.

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

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

– Individual entries, multiple or all runs, species and compartments can be selected.
– The list of species names can be sorted in either ascending or descending alphabetical order, and filtered by name.

– The list of compartments can similarly be sorted or filtered by name, and compartments can additionally be sorted by their X and Y positions on the lattice.
– The number of time points to use can be adjusted by changing the interval values of the `from`, `to` and `every` spinboxes.
– The *data units* of the model components can be set, and the *display units*, in which simulation results are to be handled and presented in plots, can be specified for timepoints, species quantities and compartment volumes. For instance, species amounts may be interpreted as either molecules, moles or concentrations, the choice determines which display units are available.
– The user can choose whether or not to average the amounts of each species in each compartment over the set of selected runs (default for stochastic simulations, hidden along with the list of runs for deterministic simulations). Averaging over many runs can approximate the deterministic outcome for systems where stochasticity is of lesser importance.
– The Dashboard displays the number of time series and surfaces, and estimate the memory requirements of each action, allowing the user to determine how quickly the action can be performed and whether the results will be comprehensible.
– The selected and rescaled datapoints can be exported from the Infobiotics Dashboard by clicking the `Export data as...` button to open a save file dialog limited to files with the extensions `.csv` (comma-separated value), `.xls` (Microsoft Excel) and `.npz` (NumPy).
– Distributions of the average quantity of each selected species at a single timepoint can be plotted as histograms for either each selected compartment over all selected runs, or each selected run over all selected compartments.
– With the time series plotting functionality, users can make exact (`combined`) or relative (`stacked/tiled`) quantitative comparisons of the temporal behaviour of multiple molecular species in multiple compartments, between several, or averaged over many, simulation runs. These plots can be exported as images for further comparison with experimental observations. Fig. 1.6 shows the time series plotting interface for the *stacked* style. When working on a stacked or tiled plot, the `Refine time series selection` button will open a dialog in which the order and visibility of subplots can be adjusted.
  When averaging over multiple runs, each line is the *sample mean* and each marker is overlaid with error bars of either the `standard deviation of the sample` (SD) or the `confidence interval` (CI) describing the accuracy of the standard deviation.
  The figure toolbar provided by Matplotlib [85] enables zooming, panning, *Subplot configuration*: adjustment of the spacing between multiple plots and the figure boundary and exporting plot image, as it appears for publication in bitmap and vector formats.
– The Infobiotics Dashboard enables users to visualise how species quantities change in time and 2D space by using 3D heat-mapped meshes or *surface* (where the vertices of the mesh correspond to model lattice points and the height of the peaks to the species quantities), to capture the distribution of each selected species over the model at a single timepoint. Multiple surfaces, one per species,

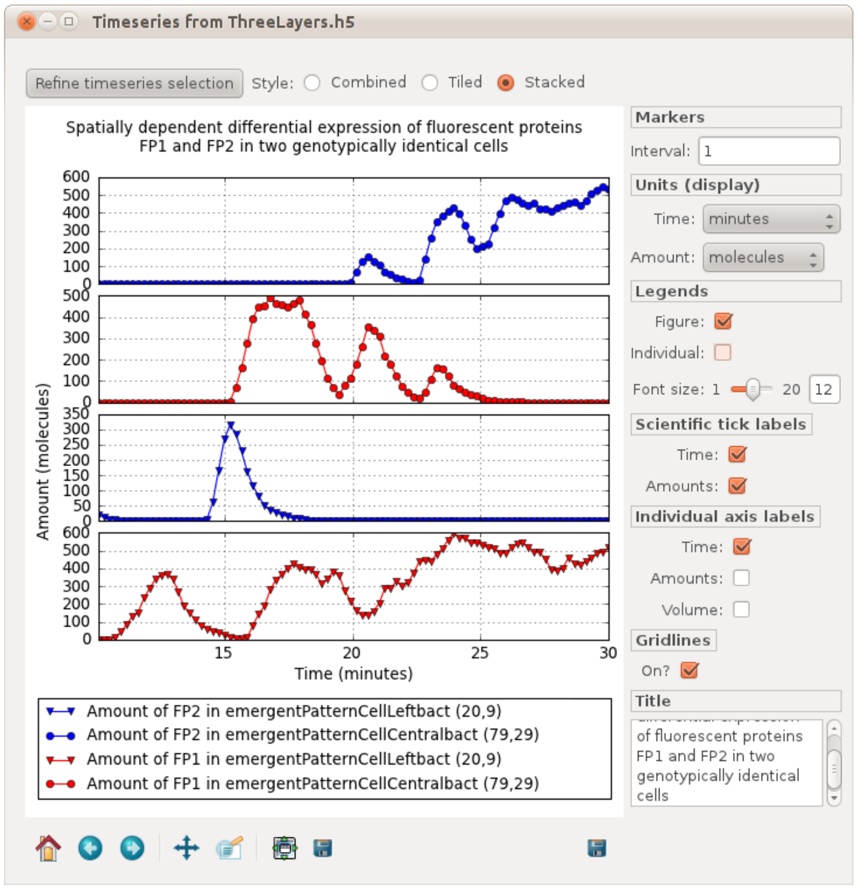

Timeseries from ThreeLayers.h5
Refine timeseries selection
Style: Combined Tiled [✓ Stacked]
Spatially dependent differential expression of fluorescent proteins FP1 and FP2 in two genotypically identical cells
Amount (molecules)
Time (minutes)
Amount of FP2 in emergentPatternCellLeftbact (20,9)
Amount of FP2 in emergentPatternCellCentralbact (79,29)
Amount of FP1 in emergentPatternCellLeftbact (20,9)
Amount of FP1 in emergentPatternCellCentralbact (79,29)
Markers
Interval: 1
Units (display)
Time: minutes
Amount: molecules
Legends
Figure: ✓
Individual:
Font size: 1 20 12
Scientific tick labels
Time: ✓
Amounts: ✓
Individual axis labels
Time: ✓
Amounts:
Volume:
Gridlines
On? ✓
Title
of fluorescent proteins FP1 and FP2 in two genotypically identical cells

Fig. 1.6: Time series: Stacked plot style.

each corresponding to particular species, can be visualized simultaneously side-by-side for qualitative comparison. The overlaid scalar bars map heat as colour to quantities.
Figure 1.7 shows an example in which two surfaces plots of 1600 compartments (40x40) are rendered. Time is progressed either manually, by dragging the time-point index slider, or automatically using the `Play/Pause` button.
Surfaces plots provide an intuitive means of qualitatively gauging the behaviour of population level models, that may (cautiously) be compared to microscopy data.

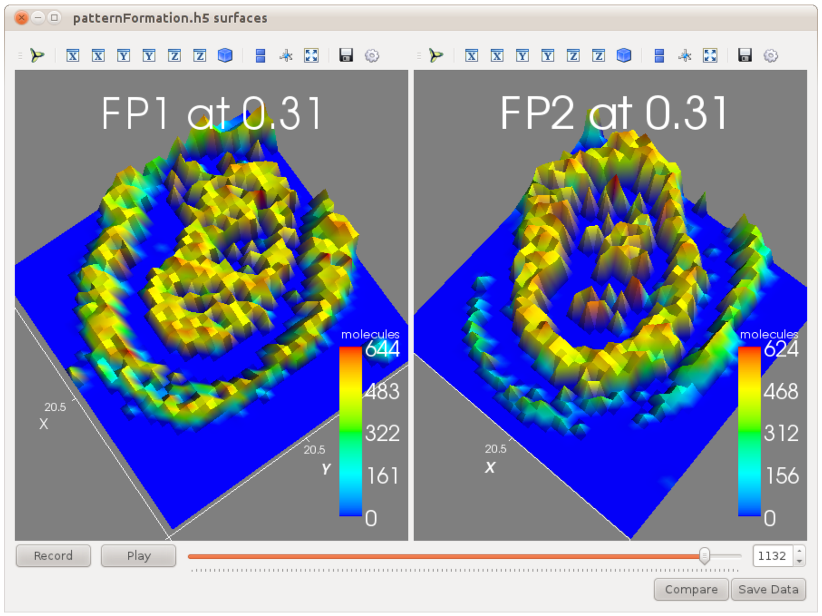


Fig. 1.7: Surface plots showing expression patterns of two fluorescent proteins.

1.4.3 Model Checking

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

Probabilistic model checking is a probabilistic variant of classical model checking augmented with quantitative information regarding the likelihood that certain transitions occur and the times which they do so. Probabilistic model checking works with *Discrete time Markov Chains (DTMCs)*, *Continuous time Markov chains (CTMCs)* or *Markov Decision Processes (MDPs)*. A continuous time Markov chain (CTMC) is defined by a set of *states*, a set of *initial* states and a *transition rate matrix* from which the rate at which a transition occurs between each pair of states is taken as a parameter of an exponential distribution. Queries which check model properties are defined as logical statements, often *probabilistic logics*: CSL *(Continuous Stochastic Logic)* [3] for CTMCs, PCTL *(Probabilistic Computation Tree Logic)* [64] for DTMCs and MDPs.

The Infobiotics Workbench is equipped with a model checking module, called PMODELCHECKER. Properties of stochastic P system models can be expressed as probabilistic logic formulas and automatically verified using third party model checking softwares, namely PRISM [75, 77] and MC2 [35, 33, 34]. PMODELCHECKER [112] extends this capability to LPP system models by acting as wrapper interface between LPP systems and the model checkers PRISM and MC2.

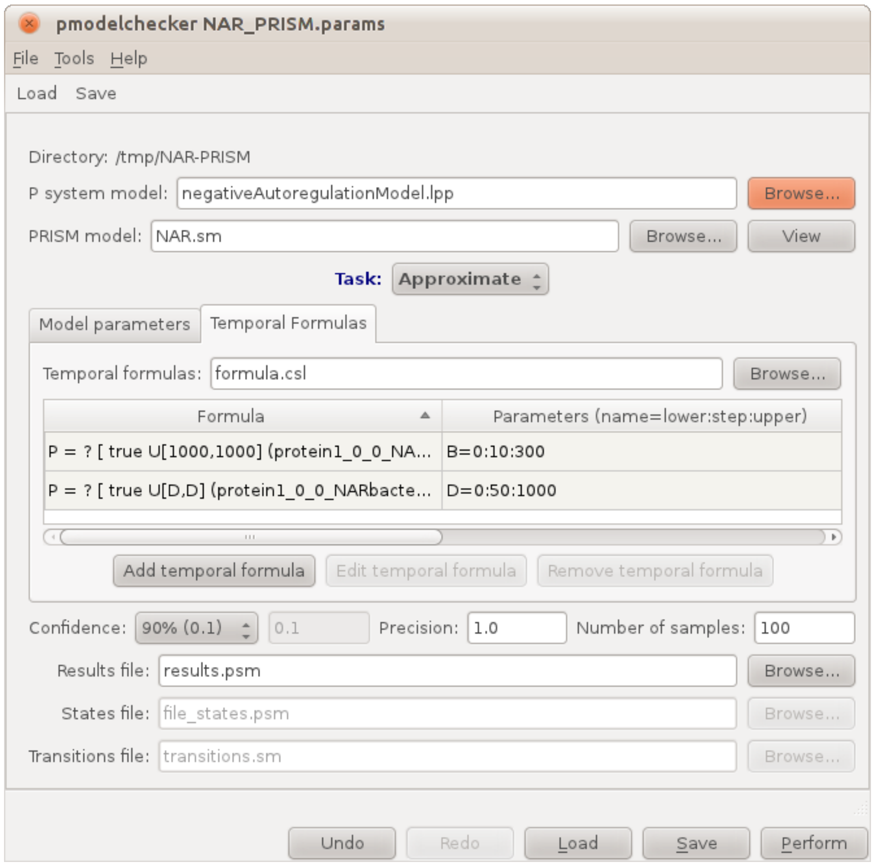
To perform probabilistic model checking with PRISM, LPP systems are loaded and automatically converted into a Reactive Modules specification (a CTMC) [2] that PRISM can accept as input. Parameters are created for the lower and upper bounds of the number of molecules of each species in each compartment: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

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

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

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

Once a model checking experiment has completed the results interface is loaded from the file specified by the `results_file` parameter. The output is the same for either model checking experiment: for each formula a list of the probability of each property being fulfilled for each combination of formula parameters, usually time plus several others (e.g. Fig. 1.9). The varying probabilities of each property can be plotted in two ways: a 2D plot of the probability that the property is satisfied against all values of one variable (Fig. 1.9a) or a 3D plot of probability against all values two variables (Fig. 1.9b), at a single value of each remaining variable. The constant values of the remaining variables can be set using sliders which are dynamically added to the results interface above the plot depending on availability and the currently selected axis variables. In this way both 2D and 3D plots can be



Directory: /tmp/NAR-PRISM

P system model: negativeAutoregulationModel.lpp

PRISM model: NAR.sm

Task: Approximate

Temporal formulas: formula.csl

Formula	Parameters (name=lower:step:upper)
P = ? [true U[1000,1000] (protein1_0_0_NA...	B=0:10:300
P = ? [true U[D,D] (protein1_0_0_NARbacte...	D=0:50:1000

Confidence: 90% (0.1) 0.1 Precision: 1.0 Number of samples: 100

Results file: results.psm

States file: file_states.psm

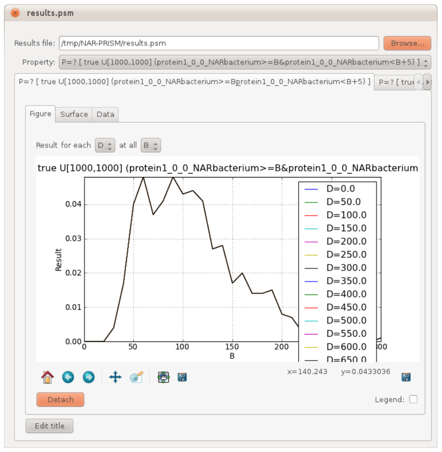
Transitions file: transitions.sm

Fig. 1.8: PMODELCHECKER parameterisation interfaces.

used to visualise queries with greater numbers of variables, enabling the results of N-dimensional queries to be interrogated in a consistent manner.

1.4.4 Optimisation

Both stochastic and deterministic models are dependent on the correct model structure and accurate rate constants to accurately reproduce cellular behaviour. Unfortunately well-characterised rate constants are in very short supply, and those that are known for some models are used as ersatz values in models of similar systems. In the scenario, where the components and interactions are known but other parameters are not, it is acceptable to try estimate the rate constants using parameter optimisation to fit model dynamics to laboratory observations.



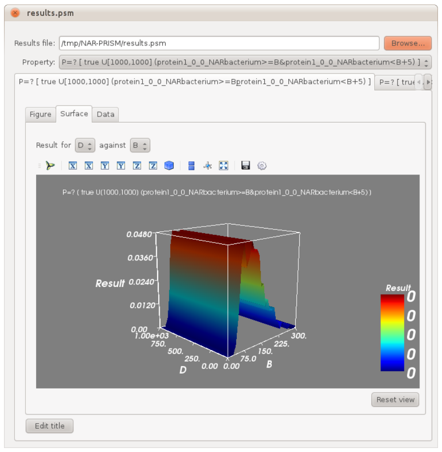
results.psm

Results file: /tmp/NAR-PRISM/results.psm

Property: P=? [true U[1000,1000] (protein1_0_0_NARbacterium>=B&protein1_0_0_NARbacterium<B+5)]

Result for each D at all B

(a) 1 variable 2D plot.



results.psm

Results file: /tmp/NAR-PRISM/results.psm

Property: P=? [true U[1000,1000] (protein1_0_0_NARbacterium>=B&protein1_0_0_NARbacterium<B+5)]

Result for D against B

P=? (true U(1000,1000) (protein1_0_0_NARbacterium>=B&protein1_0_0_NARbacterium<B+5))
Result
D
B
Figure
Surface
Data
Reset view
Edit title

(b) 2 variable 3D plot.

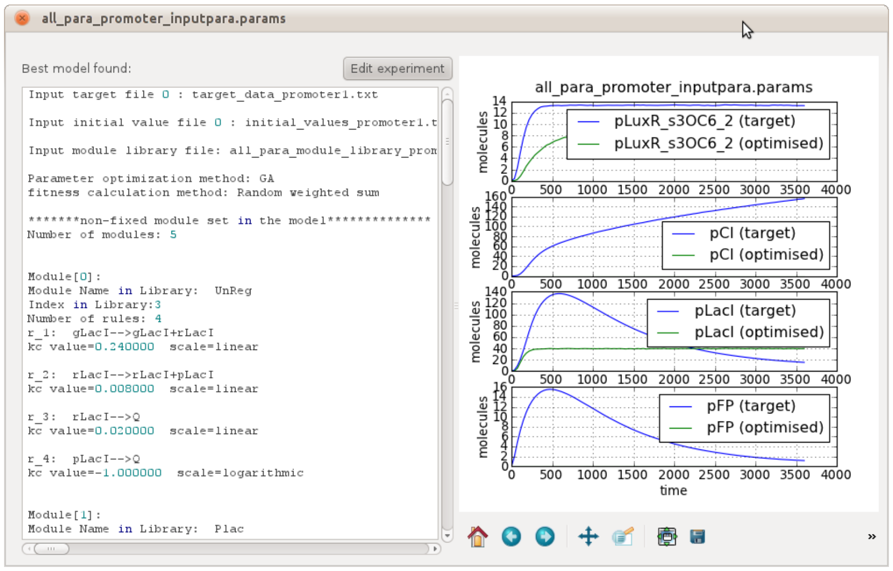
Fig. 1.9: Model checking experiments results interface.

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

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

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

POPTIMIZER searches the parameter and structure spaces of *single compartment* stochastic P systems with implementations of state-of-the-art population-based optimisation algorithms: Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [63], Estimation of Distribution Algorithms (EDA), Differential Evolution (DE) [122] and Genetic Algorithms (GA) [60]. Optimisation is limited to single compartment models, partly due to the increased complexity of algorithmically manipulating spatially distributed or hierarchically organised compartmental structures (and the distinction made between these by the LPP formalism), but more pragmatically because repeated stochastic simulation of each individual in a population of (potentially unfeasible) single compartment models (with suboptimal rate constants) is



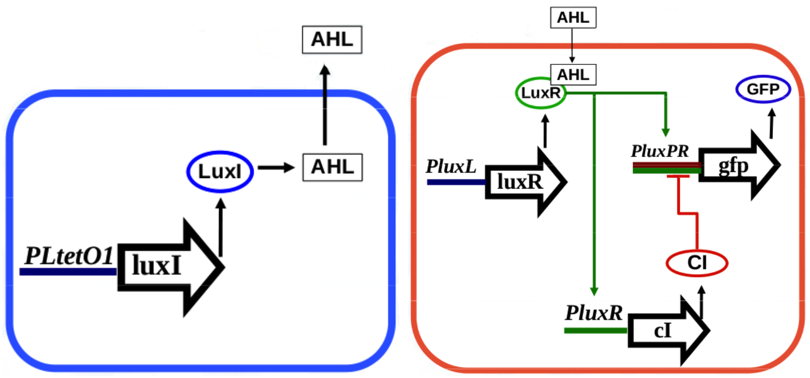
```
Input target file 0 : target_data_promoter1.txt

Input initial value file 0 : initial_values_promoter1.t

Input module library file: all_para_module_library_prom

Parameter optimization method: GA
fitness calculation method: Random weighted sum

*******non-fixed module set in the model**************
Number of modules: 5


Module[0]:
Module Name in Library:  UnReg
Index in Library:3
Number of rules: 4
r_1:  gLacI-->gLacI+rLacI
kc value=0.240000  scale=linear

r_2:  rLacI-->rLacI+pLacI
kc value=0.008000  scale=linear

r_3:  rLacI-->Q
kc value=0.020000  scale=linear

r_4:  pLacI-->Q
kc value=-1.000000  scale=logarithmic


Module[1]:
Module Name in Library:  Plac
```

Fig. 1.10: POPTIMIZER results interface.

very computationally expensive. Simulating many copies of those compartments, interacting on a 2D lattice would multiply the cost and providing suitable or accurate target data would be difficult also. Thus model optimisation is generally only tractable with smaller models (as with model verification). However, submodels can be optimised in isolation and then reintegrated, provided they can be decoupled: the assumption made by the modularised, engineering approach to synthetic biology.

POPTIMIZER uses a *nested genetic algorithm* [113, 20] to generate a set of candidate models, initially by random choice and thereafter by mutating the fittest individuals of the previous generation, performing several rounds of parameter optimisation on each individual to ensure that the candidates are given a fair chance of fitting the desired behaviour (as previous rate constants may be unsuited to the updated reaction network) before using the final fitness to select the next generation.

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



(a) Sender cell. (b) Pulsing cell.

Fig. 1.11: Two different bacterial strains of the pulse generator.

1.5 Case study

In this section, we demonstrate the use of the IBW features in a case study. Here, we select the *pulse generator* example, which consists of the synthetic bacterial colony designed by Ron Weiss' group in [5, 4]. This model implements the propagation of a wave of gene expression in a bacterial colony. For other applications of this modelling see [46, 124].

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

– Sender cells contain the gene `luxI` from *Vibrio fischeri*. This gene codifies the enzyme `LuxI` responsible for the synthesis of the molecular signal `3OC6-HSL` (`AHL`). The `luxI` gene is expressed constitutively under the regulation of the promoter `PLtetO1` from the tetracycline resistance transposon.

– Pulsing cells contain the `luxR` gene from *Vibrio fischeri* that codifies the `3OC6-HSL` receptor protein `LuxR`. This gene is under the constitutive expression of the promoter `PluxL`. It also contains the gene cI from lambda phage codifying the repressor `CI` under the regulation of the promoter `PluxR` that is activated upon binding of the transcription factor `LuxR_3OC6_2`. Finally, this bacterial strain carries the gene `gfp` that codifies the green fluorescent protein under the regulation of the synthetic promoter `PluxPR` combining the `Plux` promoter (activated by the transcription factor `LuxR_3OC6_2`) and the `PR` promoter from lambda phage (repressed by the transcription factor `CI`).

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

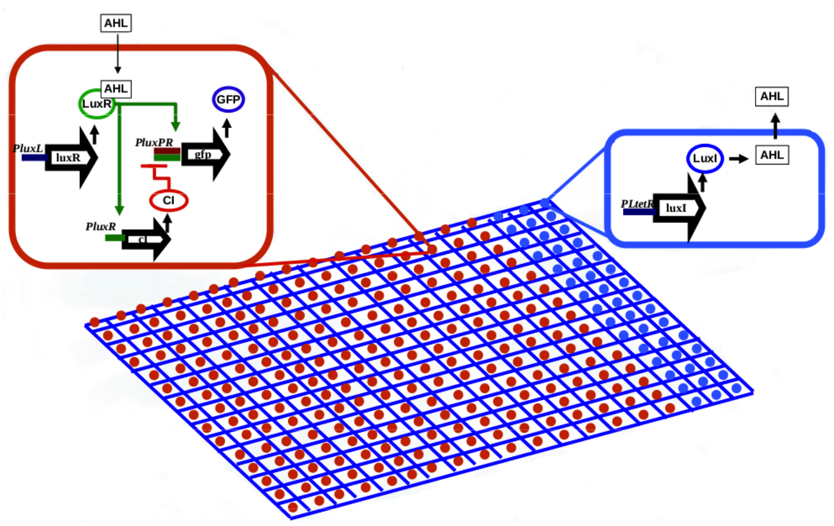


Fig. 1.12: Spatial distribution of sender and pulsing cells.

1.5.1 LPP Model

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

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

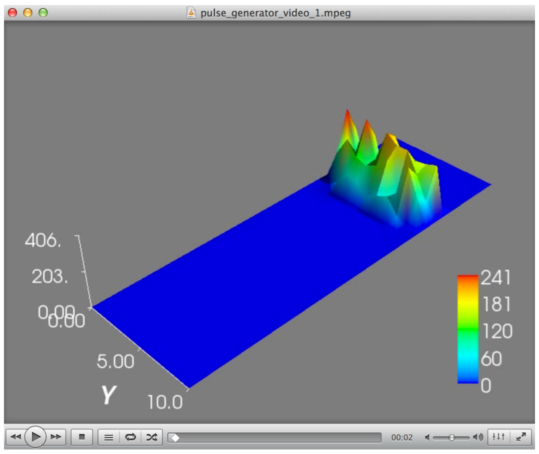


Fig. 1.13: Spatial propagation of `GFP` over the bacterial colony.

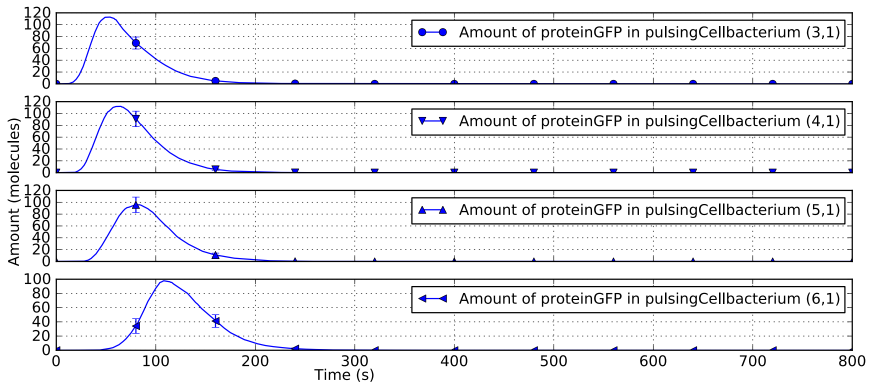


Fig. 1.14: Propagation of `GFP` over a pulsing cell.

1.5.2 Simulations

The final model has 341 compartments (11×31), 28 molecular species and 8783 rules in total. 5 stochastic simulation runs of 800 simulated seconds required an average of 2 minutes and 4 seconds wall clock time on a single 2.20GHz core of an Intel(R) Core(TM) i7-2670QM. The enhanced multi-compartmental stochastic simulation algorithm performed $65,679,239$ total reactions per run on average, achieving a rate of 528,968 reactions per second. It should be noted the time required to simulate a model is highly dependent on the structure of the reaction network in addition to the number of the compartments and reactions, and that flucutations in the number of molecules in the system as its state changes can dramatically impact the rate of simulation.

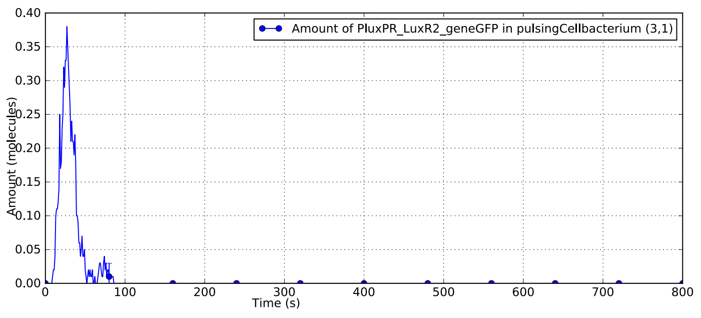


Fig. 1.15: `signal3OC6` level over time.

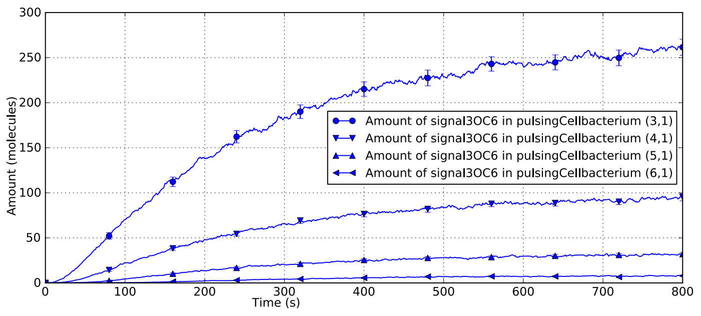


Fig. 1.16: `signal3OC6` level over time.

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

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

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

Fig. 1.16 shows the signal molecule `signal3OC6` amount over time. The figure suggests that the further away the pulsing cells are from the sender cells the less likely they are to produce a pulse.

1.5.3 Model Checking

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

PRISM

In PRISM, properties are specified in Continuous Stochastic Logic (CSL) [3] - an extension of Probabilistic Continuous Time Logic (PCTL) [64] for CTMCs.

CSL fomulas are interpreted over CTMCs. The execution of a CTMC constructs a set of *paths*, which are *infinite* sequences of states. Apart from the usual operators from classical logic such as $\wedge$ (and), $\vee$ (or) and $\Rightarrow$ (implies), CSL has the *probabilistic operator* $\mathrm{P}_{\sim r}$, where $0 \leq r \leq 1$ is a *probability bound* and $\sim \in \{<,>,\leq,\geq,=\}$[2]. Intuitively, a state, s, of a model satisfies $\mathrm{P}_{\sim r}[\varphi]$ if, and only if, the probability of taking a path from s satisfying the *path formula* φ is bounded by '$\sim r$'. The following path formulas φ are allowed: $\mathrm{X}\phi$; $\mathrm{F}\phi$; $\mathrm{G}\phi$; $\phi \mathrm{U} \psi$; and $\phi \mathrm{U}^{\leq k} \psi$ (Note that the operators $\mathrm{F}\phi$ and $\mathrm{G}\phi$ can actually be derived from $\phi \mathrm{U} \psi$).

As an example, the property that "the probability of φ eventually occurring is greater than or equal to b" can expressed in CSL as follows:

$$\mathrm{P}_{\geq b}[\texttt{true}\ \mathrm{U}\ \varphi].$$

The informal meanings of such formulas are:

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

As well as the probabilistic operator $\mathrm{P}_{\sim r}$, CSL also includes $\mathrm{S}_{\sim r}$ and $\mathrm{R}_{\sim r}$ operators to express properties regarding the *steady-state* behaviour and expected values of *rewards* respectively. There are four different types of reward formulas, which are the *reachability* reward $\mathrm{R}_{\sim r}[\mathrm{F}\varphi]$, *cumulative* reward $\mathrm{R}_{\sim r}[\mathrm{C} \leq t]$, *instantaneous* reward $\mathrm{R}_{\sim r}[\mathrm{I} = t]$ and *steady-state* reward $\mathrm{R}_{\sim r}[\mathrm{S}]$. The informal semantics of these formulas are given below [76]:

[2] The $\mathrm{P}_{\sim r}$ operator is the probabilistic counter-part of path-quantifiers $\forall$ and $\exists$ of CTL.

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

MC2

In MC2, properties are specified in a variant of Probabilistic Linear Temporal Logic (PLTL) [91] (which is a probabilistic extension of Linear Temporal Logic (LTL) [101]. This variant is called PLTL_c [36], the discrete time steps of which correspond to the logging interval of the simulation.

PLTL_c formulas are interpreted over a finite set of finite paths (e.g., simulation traces and time series). The PLTL_c language extends the syntax of LTL with *numerical constraints* and a *probability* operator. Therefore, in addition to the standard boolean operators (e.g., $\wedge$, $\vee$ and $\Rightarrow$) and temporal operators (e.g., $X\phi$, $F\phi$, $G\phi$ and $\phi U \psi$), PLTL_c includes numerical constraints in the form of *value* $\sim$ *value* ($\sim \in \{<, >, \leq, \geq, =, \neq\}$), where *value* is defined as follows [36]:

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

where `Int` denotes integer numbers, `Real` denotes real numbers, `$fVariable` denotes *free* variables, [`molecule`] denotes molecular concentrations of biochemical species, `max`[`molecule`] denotes a function which "operates over all the values of a species and returns the maximum of the species value in simulation runs" [36] and `d`[`molecule`] denotes a function which returns "the derivative of the concentration of the species at each time point" [36] in a simulation run.

PLTL_c also includes the *probabilistic operators* $P_{\sim r}$, where $0 \leq r \leq 1$ is a *probability bound* and $\sim \in \{<, >, \leq, \geq\}$ (without equality "=").

Property Patterns

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

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

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

Table 1.1: Property patterns.

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

$$\mathrm{P}_{=0.9}[\mathit{true}\ \mathrm{U}^{\leq 60}\ \mathit{molecules} \geq 100].$$

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

To facilitate property specification and therefore to increase the accessibility of useful capabilities of model checking to a wider group of users, we have developed the NLQ *(Natural Language Query)* tool, which converts *natural language queries* into their corresponding formal specification language. NLQ is based on the prototype tool introduced in [37] with extra added features and support for probabilistic logics used in the model checking module of the Infobiotics Dashboard. Using the NLQ tool, users can create a set of properties simply by manipulating a configurable form with graphical user interface elements such as drop-down lists and text fields[3].

Another important feature of the NLQ tool is that it provides users with a set of so called *property patterns* based on most frequent properties in the model checking study. Since the seminal paper of Dwyer et al. [38], there have been many developments in categorising recurring properties into specific property patterns, which can be considered as generic representations of instances of numerous properties utilised in different contexts. Indeed, [38] surveyed more than five hundred temporal properties and categorised them into a handful of property patterns. [74] extended

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

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

Table 1.2: PRISM properties.

this work to include real-time specification patterns. [61] presented a similar pattern system for probabilistic properties.

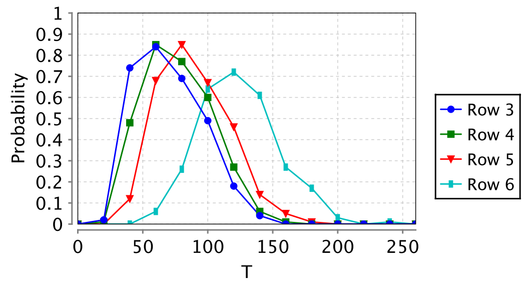
Some of the patterns used in the NLQ tool is shown in Table 1.1. These patterns provide a coherent set of templates, which guide users to construct formal expressions to represent desired properties.

Experiments

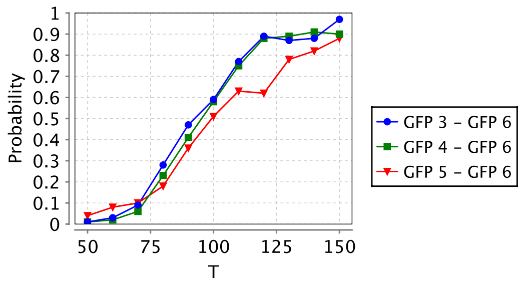
We now present the results of the probabilistic model checking experiments we carried out. Due to the well-known scalability issues that model checkers suffer we reduced the size of the lattice to 4×8, where the surrounding cells are boundary cells and 2×2-sender cells are located inside at one edge,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

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

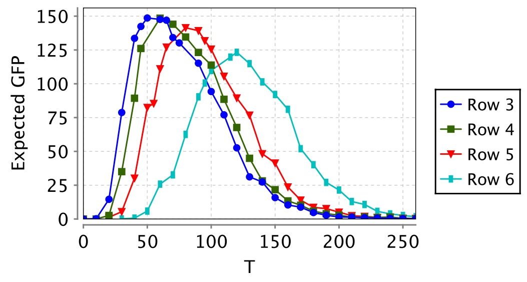
Based on these results, we have made some observations. Firstly, as Fig. 1.17a and 1.17c suggest, the GFP protein propagates through the pulsing cells. Namely, the GFP protein is first observed in the rows closer to the sender cells, then the concentration level drops until it permanently becomes zero. On the other hand, the



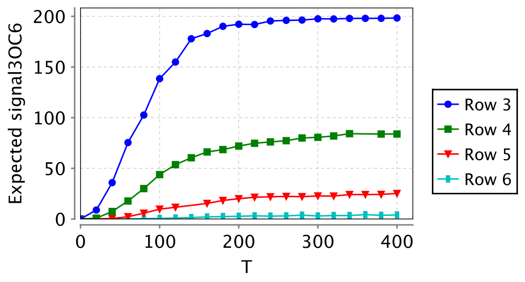
(a) Prob. of GFP exceeds threshold (Prop. 5).



(b) Prob. of relative GFP (Prop. 6).



(c) Expected GFP protein (Prop. 7).



(d) Expected signal3OC6 (Prop. 8).

Fig. 1.17: Model checking experiment results.

concentration level in the next rows shows a similar pattern with some delay, which is proportional to the distance of the row to the sender cells. This behaviour can also be observed from Prop. 1, 2, 3 and 4. Fig. 1.17d also suggests that the further away the pulsing cells are from the sender cells the less likely they are of producing a pulse. Clearly, these results are in line with the simulation results discussed previously.

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

1.5.4 Supplementary Material

The complete model and experimental results of the pulse generator example can be downloaded from the IBW website [107]. These include LPP model files, simulation parameters, simulation results, PRISM model file, model checking parameters, a list of PRISM properties and model checking experimental results. The interested readers can try running the experiments themselves. The “README.txt” file provides a detailed guidance on how to perform the same or similar experiments.

1.6 Discussions and Conclusions

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

As we have seen so far, IBW is a complex software environment combining the power and flexibility of a formal modelling framework based on stochastic P systems enhanced with a lattice-based geometry and a modular way of grouping rules. It also includes an advanced formal verification component consisting of some probabilistic and stochastic model checking tools, PRISM and MC2, together with a natural language pattern facility allowing to formulate various queries in a free style without paying attention to specific syntactic constraints. The other key component of this tool is a model structure and parameter optimisation engine. These three components are fully integrated into an environment where they smoothly communicate, models can be edited and results of various experiments are visualised according to a broad range of options.

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

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

Table 1.3: Tools comparison.

It is difficult to compare the expressiveness of the modelling languages used by these tools, as although they all use the same P systems paradigm, they implement different features - some use deterministic execution style [11], whereas many rely on probabilistic or stochastic behaviour [7, 28, 14, 23, 68]; SRSim uses strings as opposed to all the others employing multisets; some use explicitly geometric elements [68, 14] or a topology of the environment [28], but the others make use only of the membrane structure.

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

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

The results of the simulations require a form of validation, through experiments, or in depth analysis, with mathematical and/or computational instruments, complementing the simulation. Such an analysis method is the formal verification approach based on computational models [44, 43], especially model checking. So far, only IBW and, very recently, Cyto-Sim [23] support this type of analysis. In IBW this analysis is fully integrated with the rest, the translation into PRISM is automatically obtained from the specification and the queries formulated for each model are expressed using natural language patterns. Another feature of these tools that helps post-simulation analysis is the visualisation capability. This can be observed in some of these tools, MeCoSim, SRSim, MetaPlab, IBW, as being fully integrated with the other components.

Biological systems in contrast to complex engineering systems are in many cases not fully specified. At least two aspects are not always known, the kinetic rates of some interactions and the structure of certain components. These issues are overcome by employing optimisation methods for approximating the unknown aspects. MetaPlab uses such methods to approximate functions associated to rules in MP models, BioSimWare deals with parameter estimation [12] and IBW provides mechanisms for parameter estimation and model structure optimisation in the case of stochastic systems. Recently, it is reported the possibility of using similar methods for BioSimWare [10].

Complex simulations require better algorithms implementing various semantics associated to P systems models and also the use of novel technologies. In the last years there have been investigations related to the use of parallel hardware architectures for speeding-up simulations. IBW has a parallel version that distributes simulation runs over HPC clusters. BioSimWare has a version running on distributed architectures such as grid and CUDA [8]. MeCoSim/P-Lingua platform uses CUDA for PDP systems showing in certain cases significant increase in speed and new exciting research avenues [84]. However, this facility is not fully integrated in the software platform.

Acknowledgements. JB, JT, SK, NK and MG acknowledge the support provided for synthetic biology research by EPSRC ROADBLOCK project (EP/I031642/1 & EP/I031812/1), EPSRC AUDACIOUS project (EP/J004111/1) and FP7 STREP CADMAD project. JR-C acknowledges support from Cellular Computing Applications into Systems and Synthetic Biology, TIN2009-13192, and Computational Modelling and Simulation in Systems Biology, P08-TIC-04200. MG was also partially supported by the MuVet project, (CNCS UEFISCDI), grant number PN-II-ID-PCE-2011-3-0688.

Some parts of this paper are based on the first authors PhD thesis [13].

References

1. U. Alon. Network motifs: theory and experimental approaches. *Nature reviews. Genetics*, **8**, 6, (2007), 450–61.
2. R. Alur, T. Henzinger. Reactive modules. *Formal Methods in System Design*, **15**, (1999), 7–48.
3. C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Transactions on Software Engineering*, **29**, (2003), 524–541.
4. S. Basu, Y. Gerchman, C. H. Collins, F. H. Arnold, R. Weiss. A synthetic multicellular system for programmed pattern formation. *Nature*, **434**, (2005), 1130–1134.
5. S. Basu, R. Mehreja, S. Thiberge, M.-T. Chen, R. Weiss. Spatiotemporal control of gene expression with pulse-generating networks. *PNAS*, **101**, 17, (2004), 6355–6360.
6. F. Bernardini, M. Gheorghe, F. Romero-Campero, N. Walkinshaw. A hybrid approach to modelling biological systems. In *Proc. 8th Workshop on Membrane Computing*, volume 4860 of *LNCS*. Springer (2007), pages 138–159.
7. D. Besozzi, P. Cazzaniga, G. Mauri, D. Pescini. BioSimWare : A Software for the Modeling, Simulation and Analysis of Biological Systems. In *CMC 2010, LNCS 6501*. Springer (2010), pages 119–143.
8. D. Besozzi, P. Cazzaniga, G. Mauri, D. Pescini. BioSimWare: A P systems-based simulation environment for biological systems. In *11th International Conference on Membrane Computing 2010*, LNCS 6501 (2010), pages 119–143.
9. D. Besozzi, P. Cazzaniga, D. Pescini, G. Mauri. Modelling metapopulations with stochastic membrane systems. *Biosystems*, **91**, 3, (2008), 499 – 514.
10. D. Besozzi, P. Cazzaniga, D. Pescini, G. Mauri, S. Colombo, E. Martegani. The role of feedback control mechanisms on the establishment of oscillatory regimes in the Ras/cAMP/PKA pathway in S. cerevisiae. *EURASIP Journal of Bioinformatics and Systems Biology*, , 10.
11. L. Bianco, A. Castellini. Psim: A Computational Platform for Metabolic P systems. In *Workshop on Membrane Computing* (2007), pages 1–20.
12. BioSimWare. url: http://biosimware.disco.unimib.it/.

13. J. Blakes. *Infobiotics: Computer-Aided Synthetic Systems Biology*. Ph.D. thesis, School of Computer Science, University of Nottingham, UK (2012).
14. J. Blakes, J. Twycross, F. J. Romero-Campero, N. Krasnogor. The Infobiotics Workbench: an integrated in silico modelling platform for Systems and Synthetic Biology. *Bioinformatics*, **27**, 23, (2011), 3323–3324.
15. M. Calder, A. Duguid, S. Gilmore, J. Hillston. Stronger computational modelling of signalling pathways using both continuous and discrete-state methods. In *Proceedings of CMSB 2006*, volume 4210 of *LNCS* (2006), pages 63–77.
16. M. Calder, S. Gilmore, J. Hillston. Automatically deriving ODEs from process algebra models of signalling pathways. In *Proceedings of CMSB 2005*. Edinburgh, Scotland (2005), pages 204–215.
17. M. Calder, S. Gilmore, J. Hillston. Modelling the Influence of RKIP on the ERK Signalling Pathway Using the Stochastic Process Algebra PEPA. In C. Priami, A. Inglfsdttir, B. Mishra, H. Riis Nielson, editors, *Transactions on Computational Systems Biology VII*, volume 4230 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg (2006), pages 1–23.
18. M. Calder, S. Gilmore, J. Hillston, V. Vyshemirsky. Formal methods for biochemical signalling pathways. In *Formal Methods: State of the Art and New Directions*. Springer (2006), pages 185–215.
19. B. Canton, A. Labno, D. Endy. Refine and standardization of synthetic biological parts and devices. *Nature Biotechnology*, **26**, 7, (2008), 787–793.
20. H. Cao, F. J. Romero-Campero, S. Heeb, M. Cámara, N. Krasnogor. Evolving cell models for systems and synthetic biology. *Systems and synthetic biology*, **4**, 1, (2010), 55–84.
21. Y. Cao, D. T. Gillespie, L. R. Petzold. Adaptive explicit-implicit tau-leaping method with automatic tau selection. *The Journal of Chemical Physics*, **126**, 22, (2007), 224101.
22. Y. Cao, L. Petzold. Slow Scale Tau-leaping Method. *Comput. Methods. Appl. Mech. Eng.*, **197**, (2008), 43–44.
23. M. Cavaliere, T. Mazza, S. Sedwards. Statistical Model Checking of Membrane Systems with Peripheral Proteins: Quantifying the Role of Estrogen in Cellular Mitosis and DNA Damage. In P. Frisco, M. Gheorghe, M. Pérez-Jiménez, editors, *Applications of Membrane Systems to Biology*, Emergence, Complexity and Computation. Springer (2013).
24. P. Cazzaniga, D. Pescini, D. Besozzi, G. Mauri. Tau Leaping Stochastic Simulation Method in P Systems. In *Workshop on Membrane Computing, LNCS 4361*, LNCS (2006), pages 298–313.
25. F. Ciocchetta, J. Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, **410**, 33-34, (2009), 3065–3084.
26. D. Corne, P. Frisco. Dynamics of HIV infection studied with cellular automata and conformon-P systems. *Biosystems*, **3**, 91, (2008), 531–544.
27. V. Danos, J. Feret, W. Fontana, J. Krivine. Scalable Modelling of Biological Pathways. In *Asian Symposium on Programming Systems, LNCS 4807*, LNCS (2007), pages 139–157.
28. M. A. M. del Amor, I. Pérez-Hurtado, M. J. Pérez-Jiménez, A. R.-N. nez, F. Sancho-Caparrini. A formal verification algorithm for multienvironment probabilistic p systems. *International Journal of Foundations of Computer Science*, **22**, 1, (2011), 107–118.
29. L. Dematté, C. Priami, A. Romanel. The Beta Workbench: a computational tool to study the dynamics of biological systems. *Briefings in Bioinformatics*, **9**, 5, (2008), 437–49.
30. L. Dematté, C. Priami, A. Romanel. The BlenX Language: A Tutorial. In *Formal Methods for Computational Systems Biology, SFM 2008*, number 5054 in LNCS (2008), pages 123–138.
31. A. Deutsch, S. Dormann. *Cellular automata modeling of biological pattern formation*. Springer (2009).
32. M. J. Dinneen, Y.-B. Kim, R. Nicolescu. Edge- and node-disjoint paths in p systems. In M. K. G. Ciobanu, editor, *Proceedings Fourth Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2010)*, volume 40 of *EPTCS* (2010), pages 121–141.
33. R. Donaldson. MC2(PLTLc) Monte Carlo Model Checker for PLTLc properties (2008).
34. R. Donaldson, D. Gilbert. A Model Checking Approach to the Parameter Estimation of Biochemical Pathways. In *CMSB 2008, LNBI 5307*. Springer-Verlag (2008), pages 269–287.

35. R. Donaldson, D. Gilbert. A Monte Carlo Model Checker for Probabilistic LTL with Numerical Constraints. Technical report, Bioinformatics Research Centre, University of Glasgow, Glasgow (2008).
36. R. Donaldson, D. Gilbert. A monte carlo model checker for Probabilistic LTL with numerical constraints. Research Report TR-2008-282, Dept. of Computing Science, University of Glasgow (2008).
37. C. Dragomir. *From P Systems Specification to Prism*. Master's thesis, Dept. of Computer Science, University of Sheffield, Sheffield, UK (2009).
38. M. B. Dwyer, G. S. Avrunin, J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering*, ICSE '99. ACM (1999), pages 411–420.
39. J. Elf, M. Ehrenberg. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems Biology*, **1**, 2, (2004), 230–236.
40. G. B. Ermentrout, L. Edelstein-Keshet. Cellular automata approaches to biology. *Journal of Theoretical Biology*, **160**, (1993), 97–133.
41. J. R. Faeder, M. L. Blinov, W. S. Hlavacek. Rule-Based Modeling of Biochemical Systems with BioNetGen. In I. V. Maly, editor, *Methods in Molecular Biology, Systems Biology*, volume 500 of *Methods in Molecular Biology*. Humana Press, Totowa, NJ (2009), pages 113–167.
42. A. Feiglin, A. Hacohen, A. Sarusi, J. Fisher, R. Unger, Y. Ofran. Static network structure can be used to model the phenotypic effects of perturbations in regulatory networks. *Bioinformatics*, **28**, 21, (2012), 2811–2818.
43. J. Fisher, T. Henzinger. Executable Biology. In *Proceedings of the 2006 Winter Simulation Conference* (2006), pages 1675–1682.
44. J. Fisher, T. A. Henzinger. Executable cell biology. *Nature Biotechnology*, **25**, 11, (2007), 1239–1249.
45. J. Fisher, N. Piterman, E. J. A. Hubbard, M. J. Stern, D. Harel. Computational insights into caenorhabditis elegans vulval development. *PNAS*, **102**, 6, (2005), 1951–1956.
46. D. Florine, J. Santiago, K. Betz, J. Twycross, S.-Y. Park, L. Rodriguez, M. Gonzalez-Guzman, M. Jensen, N. Krasnogor, M. Holdsworth, M. Blackledge, S. Cutler, P. Rodriguez, J. Marquez. A Thermodynamic Switch Modulates Abscisic Acid Receptor Sensitivity. *EMBO Journal*, **30**, (2011), 4171–4184.
47. P. Frisco. *Computing with cells: Advances in membrane computing*. Oxford University Press (2009).
48. A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, H. Kitano. CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks. *Proceedings of the IEEE*, **96**, 8, (2008), 1254–1265.
49. M. Galassi. *GNU Scientific Library Reference Manual*. GNU, 3 edition (2009).
50. Q. Gao, F. Liu, D. Tree, D. Gilbert. Multi-cell Modelling Using Coloured Petri Nets Applied to Planar Cell Polarity. In *Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)* (2011), pages 135–150.
51. L. Gerosa. *Stochastic process algebras as design and analysis framework for synthetic biology modelling*. Master's thesis, University of Trento (2007).
52. M. Gibson, J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, **104**, 9, (2000), 1876–1889.
53. D. Gilbert, H. Fuss, X. Gu, R. Orton, S. Robinson, V. Vyshemirsky, M. J. Kurth, C. S. Downes, W. Dubitzky. Computational methodologies for modelling, analysis and simulation of signalling networks. *Brief. Bioinformatics*, **7**, (2006), 339–353.
54. D. Gilbert, M. Heiner, S. Lehrack. A unifying framework for modelling and analysing biochemical pathways using Petri nets. In *Proceedings of the 2007 international conference on Computational methods in systems biology*, CMSB'07. Springer-Verlag, Berlin, Heidelberg (2007), pages 200–216.
55. D. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, **22**, 4, (1976), 403–434.

56. D. T. Gillespie. Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry*, **81**, 25, (1977), 2340–2361.
57. D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, **188**, (1992), 404–425.
58. D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, **115**, 4, (2001), 1716.
59. D. T. Gillespie. Stochastic simulation of chemical kinetics. *Annual review of physical chemistry*, **58**, (2007), 35–55.
60. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Welsey (1989).
61. L. Grunske. Specification patterns for probabilistic quality properties. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08. ACM (2008), pages 31–40.
62. M. L. Guerriero, D. Prandi, C. Priami, P. Quaglia. Process Calculi Abstractions for Biology. Technical report, CoSBi (Center for Computational and Systems Biology), Trento, Italy (2006).
63. N. Hansen, A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computing*, **9**, (2001), 159–195.
64. H. Hansson, B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, **6**, (1994), 102–111.
65. L. H. Hartwell, J. J. Hopfield, S. Leibler, A. W. Murray. From molecular to modular cell biology. *Nature*, **402**, (1999), C47–C52.
66. M. Heiner, D. Gilbert, R. Donaldson. Petri Nets for Systems and Synthetic Biology. *Formal Methods for Computational Systems Biology*, **5016**, (2008), 215–264.
67. T. Hinze, C. Bodenstein, B. Schau, I. Heiland, S. Schuster. Chemical analog computers for clock frequency control based on P modules. In *Proceedings of the 12th international conference on Membrane Computing*, CMC'11. Springer-Verlag (2012), pages 182–202.
68. T. Hinze, T. Lenser, G. Escuela, I. Heiland, S. Schuster. Modelling Signalling Networks with Incomplete Information about Protein Activation States: A P System Framework for KaiABC Oscillator. In *Workshop on Membrane Computing, LNCS 5957*, LNCS (2010), pages 316–334.
69. S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, U. Kummer. COPASIa COmplex PAthway SImulator. *Bioinformatics*, **22**, 24, (2006), 3067–3074.
70. M. Hucka, A. Finney, J. Bornstein, M. Keating, E. Shapiro, J. Matthews, L. Kovitz, J. Schilstra, A. Funahashi, C. Doyle, H. Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. *Systems Biology*, **1**, (2004), 41–53.
71. J. Jack, A. Păun. Discrete modeling of biochemical signaling with memory enhancement. *Transactions on Computational Systems Biology*, , 11, (2009), 200–215.
72. K. Jensen, L. M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer (2009).
73. S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, **22**, 3, (1969), 437–467.
74. S. Konrad, B. Cheng. Real-time specification patterns. In *Proceedings of 27th International Conference on Software Engineering* (2005), pages 372 – 381.
75. M. Kwiatkowska, J. Heath, E. Gaffney. Simulation and verification for computational modelling of signalling pathways. In *Proceedings of the 2006 Winter Simulation Conference* (2006), pages 1666–1674.
76. M. Kwiatkowska, G. Norman, D. Parker. Using probabilistic model checking in systems biology. *ACM Sigmetrics Performance Evaluation Review*, **35**, 4, (2008), 14–21.
77. M. Z. Kwiatkowska, G. Norman, D. Parker. Probabilistic model checking in practice: case studies with PRISM. *Sigmetrics Performance Evaluation Review*, **32**, 4, (2005), 16–21.
78. J. C. Lagarias. Point lattices. *Handbook of Combinatorics*, **1**.

79. A. Mallavarapu, M. Thomson, B. Ullian, J. Gunawardena. Programming with models: Modularity and abstraction provide powerful capabilities for systems biology. *Journal of the Royal Society Interface*, **6**, (2009), 257–270.
80. V. Manca. The metabolic algorithm for P systems: Principles and applications. *Theor. Comput. Sci.*, **404**, 1-2, (2008), 142–155.
81. V. Manca. Metabolic p systems. *Scholarpedia*, **5**, 3, (2010), 9273.
82. V. Manca. *Infobiotics: Information in Biotic Systems*. Springer (2013).
83. V. Manca, L. Marchetti. Log-gain stoichiometric stepwise regression for MP systems. *International Journal of Foundations of Computer Science*, **22**, 01, (2011), 97–106.
84. M. Martínez-del Amor, I. Pérez-Hurtado, A. Gastalver-Rubio, A. Elster, M. Pérez-Jiménez. Population Dynamic P Systems on CUDA. In *Workshop on Membrane Computing, LNCS 7605*, LNCS (2012), pages 247–266–313.
85. Matplotlib. url: http://matplotlib.org.
86. MetaPlab. url: http://mplab.sci.univr.it/.
87. R. Milner. *Communicating and Mobile Systems: π-Calculus*. Cambridge University Press, Cambridge (1999).
88. I. I. Moraru, J. C. Schaff, B. M. Slepchenko, L. L. M. The virtual cell: an integrated modeling environment for experimental and computational cell biology. *Annals of the New York Academy of Sciences*, **971**, (2002), 595–596.
89. S. Natkin. *Les Reseaux de Petri Stochastiques et leur Application a lEvaluation des Systemes Informatiques*. Ph.D. thesis, CNAM, Paris, France (1980).
90. A. Obtulowicz. Generalized Gandy-Păun-Rozenberg Machines for Tile Systems and Cellular Automata. In M. Gheorghe, G. Păun, G. Rozenberg, A. Salomaa, S. Verlan, editors, *Membrane Computing*, volume 7184 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg (2012), pages 314–332.
91. Z. Ognjanović. Discrete linear-time probabilistic logics: Completeness, decidability and complexity. *Journal of Logic and Computation*, **16**, 2, (2006), 257–285.
92. G. Păun. Computing with Membranes. *Journal of Computer and System Sciences*, **61 (2000)**, 1, (2000), 108–143.
93. G. Păun, F. J. Romero-Campero. Membrane Computing as a Modeling Framework. Cellular Systems Case Studies. In *Formal Methods for Computational Systems Biology, LNCS 5016*, LNCS (2008), pages 168–214.
94. M. Pedersen, A. Phillips. Towards programming languages for genetic engineering of living cells. *Journal of the Royal Society Interface the Royal Society*, **6 Suppl 4**, April, (2009), S437–50.
95. M. Pedersen, G. D. Plotkin. A Language for Biochemical Systems : Design and Formal Specification. In C. Priami, editor, *Transactions on Computational Systems Biology XII, LNBI 5945* (2010), pages 77–145.
96. I. Pérez-Hurtado, L. Valencia, M. Pérez-Jiménez, M. Colomer, Riscos-Núñez. A general purpose software tool for simulating biological phenomena by means of P Systems. In *Proceedings 2010 IEEE Fifth International Conference BIC-TA 2010, Changsha, China* (2010), pages 637–643.
97. D. Pescini, D. Besozzi, G. Mauri, C. Zandron. Dynamic probabilistic P systems. *International Journal of Foundations of Computer Science*, **1**, 17, (2006), 183–204.
98. Petri nets tool database. url: http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html.
99. A. Phillips, L. Cardelli. A Correct Abstract Machine for the Stochastic Pi-calculus. In *Concurrent Models in Molecular Biology, BioConcur '04*, ENTCS (2004).
100. A. Phillips, L. Cardelli, G. Castagna. A Graphical Representation for Biological Processes in the Stochastic π-calculus. *Transactions in Computational Systems Biology*, **4230**, (2006), 123–152.
101. A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press (1977), pages 46–57.

102. C. Priami. Stochastic π-calculus. *Computer Journal*, **38**, 7, (1995), 578–589.
103. C. Priami. Algorithmic systems biology. *Commun. ACM*, **52**, (2009), 80–88.
104. C. Priami. Algorithmic systems biology. *Commun. ACM*, **52**, 5, (2009), 80–88.
105. C. Priami, P. Quaglia. Modelling the dynamics of biosystems. *Briefings in Bioinformatics*, **5**, 3, (2004), 259–269.
106. C. Priami, A. Regev, E. Shapiro, W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, **80**, (2001), 25–31.
107. Pulse Generator. url: http://www.infobiotic.org/models/pulseGenerator/pulseGenerator.html.
108. R. Ramaswamy, N. González-Segredo, I. F. Sbalzarini. A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks. *The Journal of chemical physics*, **130**, 24, (2009), 244104.
109. V. N. Reddy, M. N. Liebman, M. L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Comput. Biol. Med.*, **26**, (1996), 9–24.
110. V. N. Reddy, M. L. Mavrovouniotis, M. N. Liebman. Petri net representations in metabolic pathways. In *Proc Int Conf Intell Syst Mol Biol* (1993), pages 328–336.
111. A. Regev, W. Silverman, E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In *Pac Symp Biocomput 2001*, volume 26 (2001), pages 459–470.
112. F. Romero-Campero, M. Gheorghe, L. Bianco, D. Pescini, M. Pérez-Jiménez, R. Ceterchi. Towards Probabilistic Model Checking on P Systems Using PRISM. In *Membrane Computing*, volume 4361 of *LNCS*. Springer Berlin (2006), pages 477–495.
113. F. J. Romero-Campero, H. Cao, M. Camara, N. Krasnogor. Structure and parameter estimation for cell systems biology models. *Proceedings of the 10th annual conference on Genetic and Evolutionary Computation (GECCO '08)*, (2008), 331–339.
114. F. J. Romero-Campero, M. Gheorghe, G. Ciobanu, J. M. Auld, M. J. Pérez-Jiménez. Cellular modelling using P systems and process algebra. *Progress in Natural Science*, **17**, (2007), 375–383.
115. F. J. Romero-Campero, J. Twycross, M. Cámara, M. Bennett, M. Gheorghe, N. Krasnogor. Modular Assembly of Cell Systems Biology Models Using P Systems. *International Journal of Foundations of Computer Science*, **20**, 03, (2009), 427.
116. F. J. Romero-Campero, J. Twycross, M. Cámara, M. Bennett, M. Gheorghe, N. Krasnogor. Modular Assembly of Cell Systems Biology Models Using P Systems. *International Journal of Foundations of Computer Science*, **20**, 03, (2009), 427–442.
117. F. J. Romero-Campero, J. Twycross, H. Cao, J. Blakes, N. Krasnogor. A Multiscale Modeling Framework Based on P Systems. In *WMC9 2008*. Springer-Verlag Berlin (2009), pages 63–77.
118. The p-lingua web page. url: http://www.p-lingua.org.
119. R. P. Shetty, D. Endy, T. F. Knight. Engineering BioBrick vectors from BioBrick parts. *Journal of Biological Engineering*, **2**, (2008), 5.
120. A. Slepoy, A. P. Thompson, S. J. Plimpton. A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks. *The Journal of chemical physics*, **128**, 20, (2008), 205101.
121. M. W. Sneddon, J. R. Faeder, T. Emonet. Efficient modeling, simulation and coarse-grain of biological complexity with NFsim. *Nature Methods*, **8**, (2011), 177–183.
122. R. Storn, K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim*, **11**, (1997), 341–359.
123. F. J. W. Symons. Introduction to numerical Petri nets, a general graphical model of concurrent processing systems. *Australian Telecommun. Res.*, **14**, 1.
124. J. Twycross, L. R. Band, M. J. Bennett, J. R. King, N. Krasnogor. Stochastic and Deterministic Multiscale Models for Systems Biology: an Auxin-Transport Case Study. *BMC Systems Biology*, **4**, 1, (2010), 1–34.
125. S. Verlan, F. Bernardini, M. Gheorghe, M. Margenstern. Generalized communicating P systems. *Theor. Comput. Sci.*, **404**, 1-2, (2008), 170–184.

126. J. Will, M. Heiner. Petri nets in Biology, Chemistry, and Medicine. Bibliography. Technical report, Brandenbury University of Technology at Cottbus (2002).
127. S. Wolfram. *A new kind of science*. Champaign, IL: Wolfram Media (2002).
128. A. Yachie-Kinoshita, T. Nishino, H. Shimo, M. Suematsu, M. Tomita. A Metabolic Model of Human Erythrocytes: Practical Application of the E-Cell Simulation Environment. *Biomedicine and Biotechnology*, **2010**, (2010), 14 pages.