Ralph-Johan Back
Ion Petre
Erik de Vink (Eds.)

Computational Models for Cell Processes

2nd International Workshop, COMPMOD 2009
Eindhoven, the Netherlands, November 3, 2009

In conjunction with *Formal Methods 2009*

Electronic Proceedings in Theoretical Computer Science (EPTCS), volume 6
DOI: 10.4204/EPTCS.6

Preface

The second international workshop on *Computational Models for Cell Processes* (ComProc 2009) took place on November 3, 2009 at the Eindhoven University of Technology, in conjunction with *Formal Methods 2009*. The first edition of the workshop (2008) took place in Turku, Finland, in conjunction with *Formal Methods 2008*. This volume contains the final versions of all contributions accepted for presentation at the workshop.

The goal of the CompMod workshop series is to bring together researchers in computer science (especially in formal methods) and mathematics (both discrete and continuous), interested in the opportunities and the challenges of systems biology. CompMod 2009 has received 19 submissions, authored by 60 different authors from 7 countries. Most authors came from Italy (26), UK (11), and the Netherlands (11). The Program Committee has selected 8 papers for presentation at the workshop. We thank the PC members for their excellent work in making this selection. The CompMod 2009 Program Committee consisted of:

- Ralph-Johan Back (Åbo Akademi University, Finland) Co-Chair
- Lubos Brim (Masaryk University, Czech Republic)
- Muffy Calder (University of Glasgow, UK)
- Eugen Czeizler (Åbo Akademi University, Finland)
- Vincent Danos (University of Edinburgh, UK)
- Russ Harmer (University of Paris 7, France)
- Monika Heiner (Brandeburg University of Technology Cottbus, Germany)
- Jane Hillston (University of Edinburgh, UK)
- Ina Koch (Max Planck Institute for Molecular Genetics, Germany)
- Vincenzo Manca (Universita di Verona, Italy)
- Giancarlo Mauri (Universita Degli Studi di Milano-Bicocca, Italy)
- Satoru Miyano (University of Tokio, Japan)
- Andrei Păun (Louisiana Tech University, US)
- Ion Petre (Åbo Akademi University, Finland) Co-Chair
- Jaco van de Pol (University of Twente, the Netherlands)
- Alberto Policriti (Universita di Udine, Italy)
- Erik de Vink (Eindhoven University of Technology, the Netherlands) Co-Chair

The scientific program of the workshop spans an interesting mix of approaches to systems biology, ranging from quantitative to qualitative techniques, from continuous to discrete mathematics, and from deterministic to stochastic methods. We thank our three invited speakers

- Jasmin Fisher (Microsoft Research Cambridge, UK)

R.J.Back, I.Petre, E. de Vink (Eds.): Computational
Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. i–ii, doi:10.4204/EPTCS.6.0

- Hidde de Jong (INRIA Grenoble-Rhone-Alpes, France)
- Grzegorz Rozenberg (Leiden, the Netherlands; Boulder, Colorado, US)

for accepting our invitation and for presenting some of their recent results at CompMod 2009.

Similarly as last year, a special issue of *LNBI Transactions on Computational Systems Biology* (Springer) will be based on CompMod 2009. The forthcoming special issue will have a separate, open call for papers in early 2010 and all submissions will be subject to a separate review process.

This edition of CompMod was jointly organized with the *EC-MOAN* project. We have organized a special session of the workshop dedicated to the project, where a brief overview of *EC-MOAN* was presented, together with the invited talk by Hidde de Jong (a member of the project), and a contributed paper by members of *EC-MOAN*. We are grateful to Jaco van de Pol and to Lubos Brim for agreeing to collaborate with us on the workshop. We also thank FME for their financial support.

We would also like to thank the editorial board of the *Electronic Proceedings in Theoretical Computer Science* (EPTCS) for accepting to publish these proceedings in their series.

Ralph-Johan Back, Ion Petre, Erik de Vink
Turku, Finland and Eindhoven, the Netherlands, October 2009
Workshop organizers and PC co-chairs

Table of Contents

Mutual Mobile Membranes with Timers*

Bogdan Aman
Romanian Academy, Institute of Computer Science
A.I.Cuza University of Iaşi, Romania
baman@iit.tuiasi.ro

Gabriel Ciobanu
Romanian Academy, Institute of Computer Science
A.I.Cuza University of Iaşi, Romania
gabriel@info.uaic.ro

A feature of current membrane systems is the fact that objects and membranes are persistent. However, this is not true in the real world. In fact, cells and intracellular proteins have a well-defined lifetime. Inspired from these biological facts, we define a model of systems of mobile membranes in which each membrane and each object has a timer representing their lifetime. We show that systems of mutual mobile membranes with and without timers have the same computational power. An encoding of timed safe mobile ambients into systems of mutual mobile membranes with timers offers a relationship between two formalisms used in describing biological systems.

1 Introduction

Membrane systems are essentially parallel and nondeterministic computing models inspired by the compartments of eukaryotic cells and their biochemical reactions. The structure of the cell is represented by a set of hierarchically embedded regions, each one delimited by a surrounding boundary (called membrane), and all of them contained inside an external special membrane called *skin*. The molecular species (ions, proteins, etc.) floating inside cellular compartments are represented by multisets of objects described by means of symbols or strings over a given alphabet. The objects can be modified or communicated between adjacent compartments. Chemical reactions are represented by evolution rules which operate on the objects, as well as on the compartmentalized structure (by dissolving, dividing, creating, or moving membranes).

A membrane system can perform computations in the following way: starting from an initial configuration which is defined by the multiset of objects initially placed inside the membranes, the system evolves by applying the evolution rules of each membrane in a nondeterministic and maximally parallel manner. A rule is applicable when all the objects which appear in its left hand side are available in the region where the rule is placed. The maximally parallel way of using the rules means that in each step, in each region of the system, we apply a maximal multiset of rules, namely a multiset of rules such that no further rule can be added to the set. A halting configuration is reached when no rule is applicable. The result is represented by the number of objects from a specified membrane.

Several variants of membrane systems are inspired by different aspects of living cells (symport and antiport-based communication through membranes, catalytic objects, membrane charge, etc.). Their computing power and efficiency have been investigated using the approaches of formal languages and grammars, register machines and complexity theory. Membrane systems (also called P systems) are presented together with many variants and examples in [33]. Several applications of these systems are presented in [20]. An updated bibliography can be found at the P systems web page [35].

A first attempt to define mobile P systems is presented in [34] where the rules are similar to those of mobile ambients [10]. Inspired by the operations of endocytosis and exocytosis, namely moving a membrane inside a neighbouring membrane (endocytosis) and moving a membrane outside the membrane

*The research for this paper was partially supported by CNCSIS IDEI 402/2007 and CNCSIS TD 345/2008.

R.J.Back, I.Petre, E. de Vink (Eds.): Computational
Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 1–15, doi:10.4204/EPTCS.6.1

where it is placed (exocytosis), the P systems with mobile membranes are introduced in [26] as a variant of P systems with active membranes [33].

The systems of *mutual mobile membranes* represent a variant of P systems with mobile membranes in which the endocytosis and exocytosis work whenever the involved membranes "agree" on the movement; this agreement is described by using dual objects a and $\overline{a}$ in the corresponding rules. The operations governing the mobility of the systems of mutual mobile membranes are called mutual endocytosis (mutual endo), and mutual exocytosis (mutual exo).

The structure of the paper is as follows. In Section 2 we give a formal definition of the new class of mutual mobile membranes together with their biological motivation. Section 3 contains the formal definition of systems of mutual mobile membranes with timers, a variant of systems of mutual mobile membranes in which timers are attached to each object and each membrane. Section 4 contains some results which show that we do not obtain more computational power by adding timers to objects and membranes into a system of mutual mobile membranes. Section 5 presents a translation of timed safe mobile ambients into systems of mutual mobile membranes with timers. Related work, conclusion and references finalize the paper.

2 Systems of Mutual Mobile Membranes

Endocytosis and exocytosis are general terms which refer to the process by which anything is taken into or expelled from the cell through the action of vacuoles. Exocytosis involves the movement of materials out of the cytoplasm of the cell using ATP energy. In exocytosis, a vesicle (vacuole) migrates to the membrane inner surface and fuses with the cell membrane. This process of exocytosis is how the cells of glands producing proteins (enzyme and steroids) export molecules for use in other areas of the body (for example, enzymes made in the pancreas act in the small intestine). Endocytosis of large particles is called phagocytosis; in our bodies, various types of white blood cells ingest foreign particles and bacteria by phagocytosis. Endocytosis of small particles is called pinocytosis; an example of pinocytosis is the absorption of small nutrient particles into the small intestine.

Exocytosis and endocytosis operations were considered in terms of process algebra by Cardelli [11], with careful biological motivation and formulation, while in terms of membrane computing, by Cardelli and Păun [12].

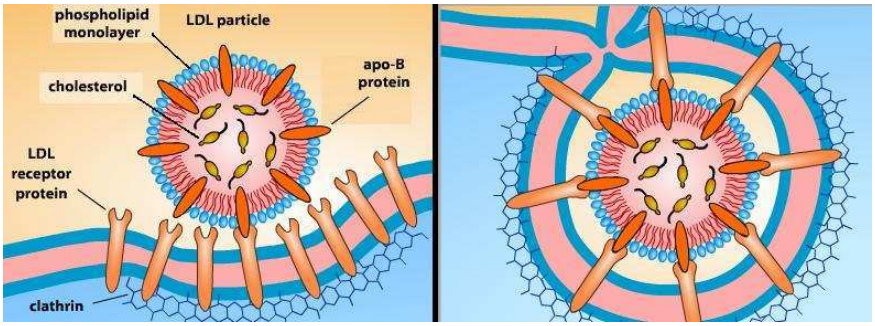


Figure 1: Receptor-Mediated Endocytosis [36]

We analyze the processes of endocytosis and exocytosis in order to define appropriate operations for mobile membranes. In *receptor-mediated endocytosis*, specific reactions at the cell surface trigger the uptake of specific molecules [36]. We present this process by an example. In such an endocytosis, a cell takes in a particle of low-density lipoprotein (LDL) from the outside. To do this, the cell uses receptors

which specifically recognize and bind to the LDL particle. An LDL particle contains one thousand or more cholesterol molecules at its core. A monolayer of phospholipids surrounds the cholesterol and it is embedded with proteins called apo-B. These apo-B proteins are specifically recognized by receptors in the cell membrane. The receptors in the coated pit bind to the apo-B proteins on the LDL particle. The pit is re-enforced by a lattice like network of proteins called clathrin. Additional clathrin molecules are then added to the lattice; eventually engulfing the LDL particle entirely.

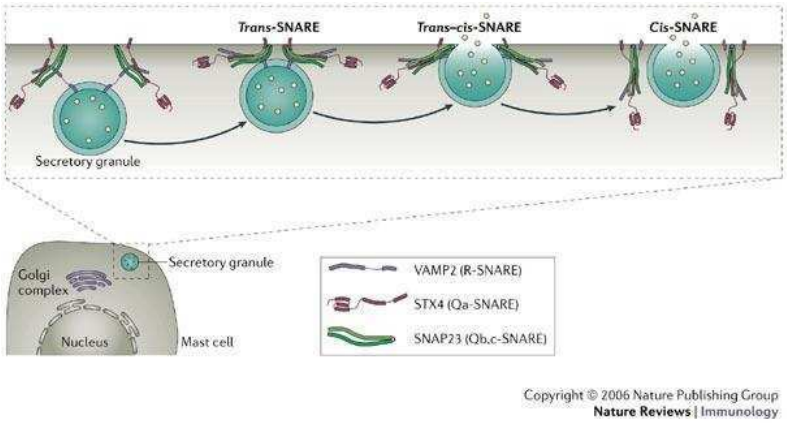


Figure 2: SNARE-Mediated Exocytosis

SNARE-mediated exocytosis is the movement of materials out of a cell via vesicles [1]. SNARES (Soluble NSF Attachment Protein Receptor)) located on the vesicles (v-SNARES) and on the target membranes (t-SNARES) interact to form a stable complex which holds the vesicle very close to the target membrane.

Endocytosis and exocytosis are modelled by mobile membranes [26]. Based on the previous examples (Figure 1 and Figure 2) where the endocytosis is triggered by the "agreement" between specific receptors and LDL particles and exocytosis by the agreement of SNARES, we introduced in [5] the *mutual* mobile membranes. In systems of mutual mobile membranes, any movement takes place only if the involved membranes agree on the movement, and this agreement is described by means of objects a and co-objects $\overline{a}$ present in the membranes involved in such a movement. An object a marks the active part of the movement, and an object $\overline{a}$ marks the passive part. The duality relation is distributive over a multiset, namely $\overline{u} = \overline{a_1} \ldots \overline{a_n}$ for $u = a_1 \ldots a_n$. The motivation for introducing the systems of mutual mobile membranes comes both from biology (e.g., receptor-mediated endocytosis), and from theoretical computer science, namely for defining models closer to the biological reality.

For an alphabet $V = \{a_1, \ldots, a_n\}$, we denote by V^* the set of all strings over V; λ denotes the empty string and $V^+ = V^* \backslash \{\lambda\}$. A multiset over V is represented by a string over V (together with all its permutations), and each string precisely identifies a multiset.

Definition 1. *A system of* $n \geq 1$ mutual mobile membranes *is a construct*

$$\prod = (V, H, \mu, w_1, \ldots, w_n, R, i_O)$$

where:

1. *V is an alphabet (its elements are called* objects*);*
2. *H is a finite set of* labels *for membranes;*
3. *$\mu \subset H \times H$ describes the* membrane structure*, such that $(i, j) \in \mu$ denotes that a membrane labelled by j is contained into a membrane labelled by i; we distinguish the external membrane (usually*

called the "skin" membrane) and several internal membranes; a membrane without any other membrane inside it is said to be elementary;

4. $w_1, \ldots, w_n \in V^*$ *are* multisets of objects *placed in the n regions of* μ*;*
5. i_O *is the output membrane;*
6. *R is a finite set of* developmental rules *of the following forms:*

mutual endocytosis

(a) $[uv]_h[\overline{u}v']_m \rightarrow [\,[w]_h w']_m$ *for* $h, m \in H, u, \overline{u} \in V^+, v, v', w, w' \in V^*$*;*
An elementary membrane labelled h enters the adjacent membrane labelled m under the control of the multisets of objects u and $\overline{u}$*. The labels h and m remain unchanged during this process; however the multisets of objects uv and* $\overline{u}v'$ *are replaced with the multisets of objects w and* w'*, respectively.*

mutual exocytosis

(b) $[\overline{u}v'[uv]_h]_m \rightarrow [w]_h[w']_m$ *for* $h, m \in H, u, \overline{u} \in V^+, v, v', w, w' \in V^*$*;*
An elementary membrane labelled h exits a membrane labelled m, under the control of the multisets of objects u and $\overline{u}$*. The labels of the two membranes remain unchanged, but the multisets of objects uv and* $\overline{u}v'$ *are replaced with the multisets of objects w and* w'*, respectively.*

The rules are applied according to the following principles:

1. All rules are applied in parallel; the rules, the membranes, and the objects are chosen nondeterministically, but in such a way that the parallelism is maximal; this means that in each step we apply a set of rules such that no further rule can be added to the set.
2. The membrane *m* from the rules of type (a) and (b) is said to be passive (identified by the use of $\overline{u}$), while the membrane *h* is said to be active (identified by the use of *u*). In any step of a computation, any object and any active membrane can be involved in one rule at most, while passive membranes are not considered to be involved in the use of the rules (hence they can be used by several rules at the same time as passive membranes).
3. When a membrane is moved across another membrane, by endocytosis or exocytosis, its whole contents (its objects) are moved.
4. If a membrane exits the system (by exocytosis), then its evolution stops.
5. All objects and membranes which do not evolve at a given step (for a given choice of rules which is maximal) are passed unchanged to the next configuration of the system.

By using the rules in this way, we can describe transitions among the configurations of the system. Some examples on how rules are applied can be found in [5].

3 Mutual Mobile Membranes with Timers

The evolution of complicated real systems frequently involves various interdependence among components. Some mathematical models of such systems combine both discrete and continuous evolutions on multiple time scales with many orders of magnitude. For example, in nature the molecular operations of a living cell can be thought of such a dynamical system. The molecular operations happen on time

scales ranging from 10^{-15} to 10^4 seconds, and proceed in ways which are dependent on populations of molecules ranging in size from as few as approximately 10^1 to approximately as many as 10^{20}. Molecular biologists have used formalisms developed in computer science (e.g. hybrid Petri nets) to get simplified models of portions of these transcription and gene regulation processes. According to [28]:

(i) "the life span of intracellular proteins varies from as short as a few minutes for mitotic cyclins, which help regulate passage through mitosis, to as long as the age of an organism for proteins in the lens of the eye."

(ii) "Most cells in multicellular organisms ... carry out a specific set of functions over periods of days to months or even the lifetime of the organism (nerve cells, for example)."

It is obvious that timers play an important role in the biological evolution. We use an example from the immune system.

Example 1 ([25])**.** *T-cell precursors arriving in the thymus from the bone marrow spend up to a week differentiating there before they enter a phase of intense proliferation. In a young adult mouse the thymus contains around* 10^8 *to* 2×10^8 *thymocytes. About* 5×10^7 *new cells are generated each day; however, only about* 10^6 *to* 2×10^6 *(roughly* $2-4\%$*) of these will leave the thymus each day as mature T cells. Despite the disparity between the numbers of T cells generated daily in the thymus and the number leaving, the thymus does not continue to grow in size or cell number. This is because approximately* 98% *of the thymocytes which develop in the thymus also die within the thymus.*

Inspired from these biological facts, we add timers to objects and membranes. We use a global clock to simulate the passage of time in a membrane system.

Definition 2. *A system of* $n\geq 1$ mutual mobile membranes with timers *is a construct*

$$\Pi=(V,H,\mu,w_1,\ldots,w_n,R,T,i_O)$$

where:

1. V, H, μ, $w_1,\ldots,w_n$, i_O *are as in Definition 1.*
2. $T\subseteq\{\Delta t \mid t\in\mathbb{N}\}$ *is a set of timers assigned to membranes and objects of the initial configuration; a timer* Δt *indicates that the resource is available only for a determined period of time* t*;*
3. R *is a finite set of* developmental rules *of the following forms:*

object time-passing

(a) $a^{\Delta t}\rightarrow a^{\Delta(t-1)}$*, for all* $a\in V$ *and* $t>0$
If an object a *has a timer* $t>0$*, then its timer is decreased.*

object dissolution

(b) $a^{\Delta 0}\rightarrow\lambda$*, for all* $a\in V$
If an object a *has its timer equal to* 0*, then the object is replaced with the empty multiset* λ*, and so simulating the degradation of proteins.*

mutual endocytosis

(c) $[u^{\Delta\widetilde{t_u}}v^{\Delta\widetilde{t_v}}]_h^{\Delta t_h}[\overline{u}^{\,\Delta\widetilde{t_{\overline{u}}}}v'^{\Delta\widetilde{t_{v'}}}]_m^{\Delta t_m}\rightarrow[\,[w^{\Delta\widetilde{t_w}}]_h^{\Delta(t_h-1)}w'^{\Delta\widetilde{t_{w'}}}]_m^{\Delta t_m}$ *for* $h,m\in H$, $u,\overline{u}\in V^+$, $v,v',w,w'\in V^*$ *and all timers are greater than* 0*;*
For a multiset of objects u*,* $\widetilde{t_u}$ *is a multiset of positive integers representing the timers of objects from* u*. An elementary membrane labelled* h *enters the adjacent membrane labelled* m *under the control of the multisets of objects* u *and* $\overline{u}$*. The labels* h *and* m *remain unchanged*

during this process; however the multisets of objects uv and $\overline{u}v'$ are replaced with the multisets of objects w and w′, respectively. If an object a from the multiset uv has the timer t_a, and is preserved in the multiset w, then its timer is now $t_a - 1$. If there is an object which appears in w but it does not appear in uv, then its timer is given according to the right hand side of the rule. Similar reasonings for the multisets $\overline{u}v'$ and w′. The timer t_h of the active membrane h is decreased, while the timer t_m of the passive membrane m remains the same in order to allow being involved in other rules.

mutual exocytosis

(d) $[\overline{u}^{\Delta\widetilde{t_{\overline{u}}}} v'^{\Delta\widetilde{t_{v'}}} [u^{\Delta\widetilde{t_u}} v^{\Delta\widetilde{t_v}}]_h^{\Delta t_h}]_m^{\Delta t_m} \to [w^{\Delta\widetilde{t_w}}]_h^{\Delta(t_h-1)} [w'^{\Delta\widetilde{t_{w'}}}]_m^{\Delta t_m}$ *for* $h,m \in H, u,\overline{u} \in V^+, v,v',w,w' \in V^*$ *and all timers are greater than* 0*;*
An elementary membrane labelled h exits a membrane labelled m, under the control of the multisets of objects u and $\overline{u}$. The labels of the two membranes remain unchanged, but the multisets of objects uv and $\overline{u}v'$ are replaced with the multisets of objects w and w′, respectively. The notations and the method of decreasing the timers are similar as for the previous rule.

membrane time-passing

(e) $[\,]_h^{\Delta t} \to [\,]_h^{\Delta(t-1)}$*, for all* $h \in H$
For each membrane which did not participate as an active membrane in a rule of type (c) *or* (d)*, if its timer is* $t > 0$*, this timer is decreased.*

membrane dissolution

(f) $[\,]_h^{\Delta 0} \to [\delta]_h^{\Delta 0}$*, for all* $h \in H$*;*
A membrane labelled h is dissolved when its timer reaches 0.

These rules are applied according to the following principles:

1. All the rules are applied in parallel: in a step, the rules are applied to all objects and to all membranes; an object can only be used by one rule and is nondeterministically chosen (there is no priority among rules), but any object which can evolve by a rule of any form, should evolve.
2. The membrane *m* from the rules of type $(c)-(d)$ is said to be passive (marked by the use of $\overline{u}$), while the membrane *h* is said to be active (marked by the use of *u*). In any step of a computation, any object and any active membrane can be involved in at most one rule, while passive membranes are not considered involved in the use of rules (c) and (d), hence they can be used by several rules (c) and (d) at the same time. Finally rule (e) is applied to passive membranes and other unused membranes; this indicates the end of a time-step.
3. When a membrane is moved across another membrane, by endocytosis or exocytosis, its whole contents (its objects) are moved.
4. If a membrane exits the system (by exocytosis), then its evolution stops.
5. An evolution rule can produce the special object δ to specify that, after the application of the rule, the membrane where the rule has been applied has to be dissolved. After dissolving a membrane, all objects and membranes previously contained in it become contained in the immediately upper membrane.
6. The skin membrane has the timer equal to ∞, so it can never be dissolved.
7. If a membrane or object has the timer equal to ∞, when applying the rules simulating the passage of time we use the equality $\infty - 1 = \infty$.

4 Mutual Mobile Membranes with and without Timers

The following results describing some relationships between systems of mutual mobile membranes with timers and systems of mutual mobile membranes without timers.

Proposition 1. *For every systems of n mutual mobile membranes without timers there exists a systems of n mutual mobile membrane with timers having the same evolution and output.*

Proof (Sketch). It is easy to prove that the systems of mutual mobile membranes with timers includes the systems of mutual mobile membranes without timers, since we can assign ∞ to all timers appearing in the membrane structure and evolution rules. □

A somehow surprising result is that mutual mobile membranes with timers can be simulated by mutual mobile membrane without timers.

Proposition 2. *For every systems of n mutual mobile membranes with timers there exists a systems of n mutual mobile membrane without timers having the same evolution and output.*

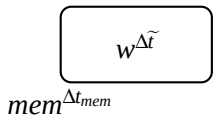
Proof. We use the notation $rhs(r)$ to denote the multisets which appear in the right hand side of a rule r. This notation is extended naturally to multisets of rules: given a multiset of rules R, the right hand side of the multiset $rhs(R)$ is obtained by adding the right hand sides of the rules in the multiset, considered with their multiplicities.

Each object $a \in V$ from a system of mutual mobile membranes with timers has a maximum lifetime (we denote it by $lifetime(a)$) which can be calculated as follows:

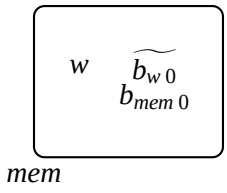
$$lifetime(a) = max(\{t \mid a^{\Delta t} \in w_i^{\widetilde{t_i}}, 1 \leq i \leq n\} \cup \{t \mid a^{\Delta t} \in rhs(R)\})$$

In what follows we present the steps which are required to build a systems of mutual mobile membranes without timers starting from a system of mutual mobile membranes with timers, such that both provide the same result and have the same number of membranes.

1. A membrane structure from a system of mutual mobile membrane with timers



   is translated into a membrane structure of a system of mutual mobile membranes without timers in the following way



   The timers of elements from a system of mutual mobile membranes with timers are simulated using some additional objects in the corresponding system of mutual mobile membranes without timers, as we show at the next steps of the translation. The object $b_{mem\,0}$ placed inside the membrane labelled *mem* is used to simulate the passage of time for the membrane. The initial multiset of objects $w^{\Delta\widetilde{t}}$ from membrane *mem* in the system of mutual mobile membranes with timers is

translated into the multiset w inside membrane *mem* in the corresponding system of mutual mobile membranes without timers together with a multiset of objects $\widetilde{b_{w\,0}}$. The multiset $\widetilde{b_{w\,0}}$ is constructed as follows: for each object $a \in w$, an object $b_{a\,0}$ is added in membrane *mem* in order to simulate the passage of time.

2. The rules $a^{\Delta t} \to a^{\Delta(t-1)}$, $a \in V$, $0 < t \leq lifetime(a)$ from the system of mutual mobile membranes with timers can be simulated in the system of mutual mobile membranes without timers using the following rules:

   $a\, b_{a\,t} \to a\, b_{a\,(t+1)}$, for all $a \in V$ and $0 \leq t \leq lifetime(a) - 1$
   The object $b_{a\,t}$ is used to keep track of the units of time t which have passed since the object a was created. This rule simulates the passage of a unit of time from the lifetime of object a in the system of mutual mobile membranes with timers, by increasing the second subscript of the object $b_{a\,t}$ in the system of mutual mobile membranes without timers.

3. The rules $a^{\Delta 0} \to \lambda$, $a \in V$ from the system of mutual mobile membranes with timers can be simulated in the system of mutual mobile membranes without timers using the following rules:

   $ab_{a\,t_a} \to \lambda$ for all $a \in V$ such that $t_a = lifetime(a)$
   If an object $b_{a\,t_a}$ has the second subscript equal with $lifetime(a)$ in the system of mutual mobile membranes without timers, it means that the timer of object a is 0 in the corresponding system of mutual mobile membranes with timers. In this case, the objects $b_{a\,t_a}$ and a are replaced by λ, thus simulating that the object a disappears together with its timer in the system of mutual mobile membranes with timers.

4. The rules $[u^{\Delta\widetilde{t_u}} v^{\Delta\widetilde{t_v}}]_h^{\Delta t_h} [\overline{u}^{\Delta\widetilde{t_{\overline{u}}}} v'^{\Delta\widetilde{t_{v'}}}]_m^{\Delta t_m} \to [\ [w^{\Delta\widetilde{t_w}}]_h^{\Delta(t_h-1)} w'^{\Delta\widetilde{t_{w'}}}]_m^{\Delta t_m}$, $h,m \in H, u,\overline{u} \in V^+, v,v',w,w' \in V^*$ with all the timers greater than 0, from the system of mutual mobile membranes with timers can be simulated in the system of mutual mobile membranes without timers using the following rules:

   $[u\, \widetilde{b_{u\,t1}} v\, \widetilde{b_{v\,t2}}\, b_{h\,t3}]_h [\overline{u}\, \widetilde{b_{\overline{u}\,t4}}\, v'\widetilde{b_{v'\,t5}}\, b_{h\,t6}]_m \to [\ [w\, \widetilde{b_{w\,t7}} b_{h\,(t3+1)}]_h w\, \widetilde{b_{\overline{w'}\,t8}}\, b_{h\,(t6+1)}]_m$, where $h,m \in H, u,\overline{u} \in V^+, v,v',w,w' \in V^*$ and for each b_{aj} we have that $0 \leq j \leq lifetime(a) - 1$.
   A multiset $\widetilde{b_{u\,t1}}$ consists of all objects $b_{a\,j}$, where a is an object from the multiset u. If an object a from the multiset uv has its timer t_a and it appears in the multiset w, then its timer becomes $t_a - 1$. If there is an object which appears in w but it is not in uv, then its timer is given according to the right hand side of the rule. Similar reasonings are also true for the multisets $\overline{u}v'$ and w'. The timer of the active membrane h is increased (object $b_{h\,t3}$ is replaced by $b_{h\,(t3+1)}$), while the timer of the passive membrane m remains the same in order to allow being used in other rules.

5. The rules $[\overline{u}^{\Delta\widetilde{t_{\overline{u}}}} v'^{\Delta\widetilde{t_{v'}}} [u^{\Delta\widetilde{t_u}} v^{\Delta\widetilde{t_v}}]_h^{\Delta t_h}]_m^{\Delta t_m} \to [w^{\Delta\widetilde{t_w}}]_h^{\Delta(t_h-1)} [w'^{\Delta\widetilde{t_{w'}}}]_m^{\Delta t_m}$, $h,m \in H, u,\overline{u} \in V^+, v,v',w,w' \in V^*$ with all the timers greater than 0, from the system of mutual mobile membranes with timers can be simulated in the system of mutual mobile membranes without timers using the following rules:

   $[\overline{u}\, \widetilde{b_{\overline{u}\,t4}}\, v'\widetilde{b_{v'\,t5}}\, b_{h\,t6} [u\, \widetilde{b_{u\,t1}} v\, \widetilde{b_{v\,t2}}\, b_{h\,t3}]_h]_m \to [w\, \widetilde{b_{w\,t7}} b_{h\,(t3+1)}]_h [w\, \widetilde{b_{\overline{w'}\,t8}}\, b_{h\,(t6+1)}]_m$, where $h,m \in H, u,\overline{u} \in V^+, v,v',w,w' \in V^*$ and for each b_{aj} we have that $0 \leq j \leq lifetime(a) - 1$.
   The way these rules work is similar to the previous case.

6. The rules $[\ ]_h^{\Delta t} \to [\ ]_h^{\Delta(t-1)}$ from the system of mutual mobile membranes with timers can be simulated in the system of mutual mobile membranes without timers using the following rules:

$b_{h\,t} \to b_{h\,(t+1)}$ for all $h \in H$ and $0 \leq t \leq t_h - 1$.
For a membrane h from the system of mutual mobile membranes with timers, t_h represents its lifetime. The object $b_{h\,t}$ is used to keep track of the units of time t which have passed from the lifetime of the membrane h. This rule simulates the passage of a unit of time from the lifetime of membrane h in the system of mutual mobile membranes with timers, by increasing the second subscript of the object $b_{h\,t}$ in the system of mutual mobile membranes without timers.

7. The rules $[\,]_h^{\Delta 0} \to [\delta]_h^{\Delta 0}$ from the system of mutual mobile membranes with timers can be simulated in the system of mutual mobile membranes without timers using the following rules:

   $[b_{h\,t}]_h \to [\delta]_h$ for all $h \in H$ such that $t = t_h$
   If an object $b_{h\,t}$ has the second subscript equal with t_h in the system of mutual mobile membranes without timers, it means that the timer of membrane h is 0 in the corresponding system of mutual mobile membranes with timers. In this case, the object $b_{h\,t}$ is replaced by δ, thus marking the membrane for dissolution and simulating that the membrane is dissolved together with its timer in the system of mutual mobile membranes with timers.

□

We are now able to prove the computational power of systems of mutual mobile membranes with timers. We denote by $\mathbb{N}tMM_m(\textit{mutual endo}, \textit{mutual exo})$ the family of sets of natural numbers generated by systems of $m \geq 1$ mutual mobile membranes with timers by using mutual endocytosis and mutual exocytosis rules. We also denote by $\mathbb{N}RE$ the family of all sets of natural numbers generated by arbitrary grammars.

Proposition 3. $\mathbb{N}tMM_3(\textit{mutual endo}, \textit{mutual exo}) = \mathbb{N}RE$.

Proof (Sketch). Since the output of each system of mutual mobile membranes with timers can be obtained by a system of mutual mobile membranes without timers, we cannot get more than the computability power of mutual mobile membranes without timers. Therefore, according to Theorem 3 from [5], we have that the family $\mathbb{N}tMM_3$ of sets of natural numbers generated by systems of mutual mobile membranes with timers is the same as the family $\mathbb{N}RE$ of sets of natural number generated by arbitrary grammars. □

5 From Timed Mobile Ambients to Mobile Membranes with Timers

A translation of safe mobile ambients into mobile membranes is presented in [19], providing also an operational correspondence between these two formalisms such that every step in safe mobile ambients is translated into a series of well-defined steps of mobile membranes. Since an extension with time for mobile ambients already exists [2, 3, 4], and one for mobile membranes is presented in this paper, it is natural to study what is the relationship between these two extensions: timed safe mobile ambients and systems of mutual mobile membranes with timers.

5.1 Timed Safe Mobile Ambients

Ambient calculus is a formalism introduced in [10] for describing distributed and mobile computation. In contrast with other formalisms for mobile processes such as the π-calculus [29] whose computational

model is based on the notion of *communication*, the ambient calculus is based on the notion of *movement*. An ambient represents a unit of movement. Ambient mobility is controlled by the capabilities *in*, *out*, and *open*. Capabilities are similar to prefixes in CCS and π-calculus [29]. Several variants of the ambient calculus have been proposed by adding and/or removing features of the original calculus [8, 23, 27]. Time has been considered in the framework of ambient calculus in [2, 3, 4].

We use $\mathscr{P}$ to denote the set of timed safe mobile ambients; m,n for *ambient names*; a,p for *ambient tags* (a stands for *active* ambients, while p stands for *passive* ambients), and ρ as a generic notation for both tags. We write $n^{\Delta t}[P]^{\rho}$ to denote an ambient having the timer Δt and the tag ρ; the tag ρ indicates that an ambient is active or passive. An ambient $n^{\Delta t}[P]^{\rho}$ represents a bounded place labelled by n in which a process P is executed.

The syntax of the timed safe mobile ambients is defined in Table 1. Process $\mathbf{0}$ is an inactive process (it does nothing). A movement $C^{\Delta t}.P$ is provided by the capability $C^{\Delta t}$, followed by the execution of P. $P \mid Q$ is a parallel composition of processes P and Q.

Table 1: Syntax of Timed Safe Mobile Ambients

$n,m,\ldots$		names	P,Q	$::=$	processes
C	$::=$	capabilities		$\mathbf{0}$	inactivity
	in n	can enter an ambient n		$C^{\Delta t}.P$	movement
	out n	can exit an ambient n		$n^{\Delta t}[P]^{\rho}$	ambient
	$\overline{\textit{in } n}$	allows an ambient n to enter		$P \mid Q$	composition
	$\overline{\textit{out } n}$	allows an ambient n to exit			

In timed safe mobile ambients the capabilities and ambients are used as temporal resources; if nothing happens in a predefined interval of time, the waiting process goes to another state. The timer Δt of each temporal resource indicates that the resource is available only for a determined period of time t. If $t > 0$, the ambient behaves exactly as in untimed safe mobile ambients. When the timer Δt expires ($t = 0$), the ambient n is dissolved and the process P is released in the surrounding parent ambient. When we initially describe the ambients, we consider that all ambients are active, and associate the tag a to them.

The passage of time is described by the discrete time progress functions ϕ_{Δ} defined over the set $\mathscr{P}$ of timed processes. This function modifies a process accordingly with the passage of time; all the possible actions are performed at every tick of a universal clock. The function ϕ_{Δ} is inspired from [7] and [21], and it affects the ambients and the capabilities which are not consumed. The consumed capabilities and ambients disappear together with their timers. If a capability or ambient has the timer equal to ∞ (i.e., simulating the behaviour of an untimed capability or ambient), we use the equality $\infty - 1 = \infty$ when applying the function ϕ_{Δ}. Another property of the time progress function ϕ_{Δ} is that the passive ambients can become active at the next unit of time in order to participate to other reductions.

For the process $C^{\Delta t}.P$ the timers of P are activated only after the consumption of capability $C^{\Delta t}$ (in at most t units of time). Reduction rules (Table 2) show how the time progress function ϕ_{Δ} is used.

Definition 3. *(Global time progress function) We define $\phi_{\Delta} : \mathscr{P} \to \mathscr{P}$, by:*

$$\phi_{\Delta}(P) = \begin{cases} C^{\Delta(t-1)}.R & \text{if } P = C^{\Delta t}.R,\ t > 0 \\ R & \text{if } P = C^{\Delta t}.R,\ t = 0 \\ \phi_{\Delta}(R) \mid \phi_{\Delta}(Q) & \text{if } P = R \mid Q \\ n^{\Delta(t-1)}[\phi_{\Delta}(R)]^{a} & \text{if } P = n^{\Delta t}[R]^{\rho},\ t > 0 \\ R & \text{if } P = n^{\Delta t}[R]^{\rho},\ t = 0 \\ P & \text{if } P = \mathbf{0} \end{cases}$$

Processes can be grouped into equivalence classes by an equivalence relation Ξ called structural congruence which provides a way of rearranging expressions so that interacting parts can be brought together. We denote by $\not\dashrightarrow$ the fact that none of the rules from Table 2, except the rule **(R-TimePass)** can be applied. The evolution of timed safe mobile ambients is given by the following reduction rules:

Table 2: Reduction rules

(R-In) $$\frac{-}{n^{\Delta t1}[in^{\Delta t2}m.P \mid Q]^a \mid m^{\Delta t3}[\overline{in\ m}^{\Delta t4}.R]^\rho \dashrightarrow m^{\Delta t3}[n^{\Delta t1}[P \mid Q]^p \mid R]^\rho}$$

(R-Out) $$\frac{-}{m^{\Delta t3}[n^{\Delta t1}[out^{\Delta t2}m.P \mid Q]^a \mid \overline{out\ m}^{\Delta t4}.R]^\rho \dashrightarrow n^{\Delta t1}[P \mid Q]^p \mid m^{\Delta t3}[R]^\rho}$$

(R-Amb) $$\frac{P \dashrightarrow Q}{n^{\Delta t}[P]^\rho \dashrightarrow n^{\Delta t}[Q]^\rho}$$ **(R-Par1)** $$\frac{P \dashrightarrow Q}{P \mid R \dashrightarrow Q \mid R}$$

(R-Par2) $$\frac{P \dashrightarrow Q,\ P' \dashrightarrow Q'}{P \mid P' \dashrightarrow Q \mid Q'}$$ **(R-Struct)** $$\frac{P' \Xi P,\ P \dashrightarrow Q,\ Q \Xi Q'}{P' \dashrightarrow Q'}$$

(R-TimePass) $$\frac{M \not\dashrightarrow}{M \dashrightarrow \phi_\Delta(M)}$$

In the rules **(R-In)**, **(R-Out)** ambient m can be *passive* or *active*, while the ambient n is *active*. The difference between *passive* and *active* ambients is that the *passive* ambients can be used in several reductions in a unit of time, while the *active* ambients can be used in at most one reduction in a unit of time, by consuming their capabilities. In the rules **(R-In)** and **(R-Out)** the *active* ambient n becomes *passive*, forcing it to consume only one capability in one unit of time. The ambients which are tagged as *passive* become *active* again by applying the global time function **(R-TimePass)**.

In timed safe mobile ambients, if a process evolves by one of the rules **(R-In)**, **(R-Out)**, while another one does not perform any reduction, then rule **(R-Par1)** should be applied. If more than one process evolves in parallel by applying one of the rules **(R-In)**, **(R-Out)**, then the rule **(R-Par2)** should be applied. We use the rule **(R-Par2)** to compose processes which are active, and the rule **(R-Par1)** to compose processes which are active and passive.

5.2 Translation

We denote by $\mathscr{M}(\Pi)$ the set of configurations obtained along all the possible evolution of a system Π of mutual mobile membranes with timers.

Definition 4. *For a system Π of mutual mobile membranes with timers, if M and N are two configurations from $\mathscr{M}(\Pi)$, we say that M reduces to N (denoted by $M \to N$) if there exists a rule in the set R of Π, applicable to configuration M such that we can obtain configuration N.*

In order to give a formal encoding of timed safe mobile ambients into the systems of mutual mobile membranes with timers, we define the following function:

Definition 5. *A translation $\mathscr{T} : \mathscr{P} \to \mathscr{M}(\Pi)$ is given by:*

$$\mathscr{T}(A) = \begin{cases} C^{\Delta t}\mathscr{T}(A_1) & \text{if } A = C^{\Delta t}.A_1 \\ [\ \mathscr{T}_1(A_1)\ ]_n^{\Delta t} & \text{if } A = n^{\Delta t}[\ A_1\ ]^\rho \\ \mathscr{T}_1(A_1)\mathscr{T}_1(A_2) & \text{if } A = A_1 \mid A_2 \\ \lambda & \text{if } A = \mathbf{0} \end{cases}$$

where the system Π *of mutual mobile membranes with timers is constructed as follows:*

$$\Pi = (V, H, \mu, w_1, \ldots, w_n, R, T, i_O)$$

as follows:

- $n \geq 1$ *is the number of ambients from A;*
- V *is an alphabet containing the C objects from* $\mathscr{T}(P)$*;*
- H *is a finite set of labels containing the labels of ambients from A;*
- $\mu \subset H \times H$ *describes the membrane structure, which is similar with the ambient structure of A;*
- $w_i \in V^*$, $1 \leq i \leq n$ *are multisets of objects which contain the C objects from* $\mathscr{T}(A)$ *placed inside membrane i;*
- $T \subseteq \{\Delta t \mid t \in \mathbb{N}\}$ *is a multiset of timers assigned to each membrane and object; the timer of each ambient or capability from A is the same in the corresponding translated membrane or object;*
- i_O *is the output membrane - can be any membrane;*
- R *is a finite set of* developmental rules*, of the following forms:*
  1. $[in^{\Delta t2} m]_n^{\Delta t1} \mid [\overline{in\ m}^{\Delta t4}]_m^{\Delta t3} \rightarrow [[\,]_n^{\Delta t1}]_m^{\Delta t3}$*, for all* $n, m \in H$ *and all in* $m, \overline{in\ m} \in V$
  2. $[[out^{\Delta t2} m]_n^{\Delta t1} \mid \overline{out\ m}^{\Delta t4}]_m^{\Delta t3} \rightarrow [\,]_n^{\Delta t1} \mid [\,]_m^{\Delta t3}$*, for all* $n, m \in H$ *and all out* $m, \overline{out\ m} \in V$

When applying the translation function we do not take into account the tag ρ, since in mobile membranes a membrane is active or passive depending on the rules which are applied in an evolution step and we do not need something similar to ambients tags.

Proposition 4. *If* P *is a timed safe mobile ambient such that* $P \rightarrow Q$*, then there exists a system* Π *of mutual mobile membranes with timers and two configurations* $M, N \in \mathscr{M}(\Pi)$*, such that* $M = \mathscr{T}(P)$*,* $M \rightarrow N$ *and* $N = \mathscr{T}(Q)$*.*

Proof (Sketch). The construction of Π is done following similar steps as in Definition 5.

If $P \dashrightarrow Q$, then there exists a rule in the set of rules R of Π such that $M \rightarrow N$ and $N = \mathscr{T}(Q)$. □

Proposition 5. *If* P *is a timed safe mobile ambient,* Π *is a system of mutual mobile membranes with timers and* $M, N \in \mathscr{M}(\Pi)$ *are two configurations, with* $M = \mathscr{T}(P)$ *and* $M \rightarrow N$*, then there exists a timed safe mobile ambient* Q *such that* $N = \mathscr{T}(Q)$*.*

Proof (Sketch). The system Π of mutual mobile membranes with timers is constructed in the same way as in Definition 5. If $M \rightarrow N$ in the Π system of mutual mobile membranes with timers, then there exist a timed safe mobile ambient Q such that $N = \mathscr{T}(Q)$. □

Remark 1. *In Proposition 5 it is possible to have* $P \not\dashrightarrow Q$*. Let us suppose that* $P = n^{\Delta t4}[in^{\Delta t1} m.in^{\Delta t2} k.out^{\Delta t3} s]^{\rho} \mid m^{\Delta t6}[\overline{in\ m}^{\Delta t5}]^{\rho}$*. By translation we obtain* $M = [in^{\Delta t1} m\ in^{\Delta t2} k\ out^{\Delta t3} s]_n^{\Delta t4}\ [\overline{in\ m}^{\Delta t5}]_m^{\Delta t6}$*. By constructing a system* Π *of mutual mobile membrane with timers as shown in Definition 5, we have that* $M, N \in \mathscr{M}(\Pi)$ *with* $M \rightarrow N$ *and* $N = [[in^{\Delta t2} k\ out^{\Delta t3} s]_n^{\Delta t4}]_m^{\Delta t6}$*. For this* N *there exists a* $Q = m^{\Delta t6}[n^{\Delta t4}[out^{\Delta t3} s.in^{\Delta t2} k]^{\rho}]^{\rho}$ *such that* $N = \mathscr{T}(Q)$ *but* $P \not\dashrightarrow Q$*.*

6 Related Work

There are some papers using time in the context of membrane computing. However time is defined and used in a different manner than in this paper. In [15] a timed P system is introduced by associating to each rule a natural number representing the time of its execution. Then a P system which always produces the same result, independently from the execution times of the rules, is called a time-independent P systems. The notion of time-independent P systems tries to capture the class of systems which are robust against the environment influences over the execution time of the rules of the system. Other types of time-free systems are considered in [13, 16].

Time of the rules execution is stochastically determined in [14]. Experiments on the reliability of the computations have been considered, and links with the idea of time-free systems are also discussed.

Time can also be used to "control" the computation, for instance by appropriate changes in the execution times of the rules during a computation, and this possibility has been considered in [18]. Moreover, timed P automata have been proposed and investigated in [6], where ideas from timed automata have been incorporated into timed P systems.

Frequency P systems has been introduced and investigated in [30]. In frequency P systems each membrane is clocked independently from the others, and each membrane operates at a certain frequency which could change during the execution. Dynamics of such systems have been investigated.

If one supposes the existence of two scales of time (an external time of the user, and an internal time of the device), then it is possible to implement accelerated computing devices which can have more computational power than Turing machines. This approach has been used in [9] to construct accelerated P systems where acceleration is obtained by either decreasing the size of the reactors or by speeding-up the communication channels.

In [17, 24] the time of occurrence of certain events is used to compute numbers. If specific events (such as the use of certain rules, the entering/exit of certain objects into/from the system) can be freely chosen, then it is easy to obtain computational completeness results. However, if the length (number of steps) are considered as result of the computation, non-universal systems can be obtained.

In [24, 31, 32] time is considered as the result of the computation by using special "observable" configurations taken in regular sets (with the time elapsed between such configurations considered as output). In particular, in [24, 31] P systems with symport and antiport rules are considered for proving universality results, and in [32] this idea is applied to P systems with proteins embedded on the membranes.

The authors of the current paper have also considered time to "control" the computation in two other formalisms: mobile ambients [2, 3, 4] and distributed π-calculus [21, 22]. Timers define timeouts for various resources, making them available only for a determined period of time. The passage of time is given by a discrete global time progress function.

7 Conclusion

We introduce a new class of mobile membranes, namely the mobile membranes with timers. Timers are assigned to each membrane and to each object. This new feature is inspired from biology where cells and intracellular proteins have a well defined lifetime. In order to simulate the passage of time, we use rules of the form $a^{\Delta t} \to a^{\Delta(t-1)}$ for objects, and $[\,]_i^{\Delta t} \to [\,]_i^{\Delta(t-1)}$ for membranes. If the timer of an object reaches 0 then the object is consumed by applying a rule of the form $a^{\Delta 0} \to \lambda$, while if the timer of a membrane i reaches 0 then the membrane is marked for dissolution by applying a rule of the form $[\,]_i^{\Delta 0} \to [\delta]_i^{\Delta 0}$.

After dissolving a membrane, all objects and membranes previously contained in it become elements of the immediately upper membrane.

We do not obtain a more powerful formalism by adding timers to objects and to membranes into a system of mutual mobile membranes. According to Proposition 1, Proposition 2 and Proposition 3, systems of mutual mobile membranes with timers and systems of mutual mobile membranes without timers have the same computational power.

In order to relate the new class to some known formalism involving mobility and time, we give a translation of timed safe mobile ambients into systems of mutual mobile membranes with timers. This encoding shows that the class of mutual mobile membranes with timers is a powerful formalism. Such a result is related to a previous one presented in [19], where it is proved an operational correspondence between the safe mobile ambients and the systems of mutual mobile membranes.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. *Molecular Biology of the Cell - Fifth Edition*. Garland Science, Taylor & Francis Group, 2008.

[2] B. Aman, G. Ciobanu. Timers and Proximities for Mobile Ambients. *Lecture Notes in Computer Science*, vol.4649, 33–43, 2007.

[3] B. Aman, G. Ciobanu. Mobile Ambients with Timers and Types. *Lecture Notes in Computer Science*, vol.4711, 50–63, 2007.

[4] B. Aman, G. Ciobanu. Timed Mobile Ambients for Network Protocols. *Lecture Notes in Computer Science*, vol.5048, 234–250, 2008.

[5] B. Aman, G. Ciobanu. Turing Completeness Using Three Mobile Membranes. *Lecture Notes in Computer Science*, vol.5715, 42–55, 2009.

[6] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, L. Tesei. Timed P Automata. *Electronic Notes in Theoretical Computer Science*, vol.227, 21–36, 2009.

[7] M. Berger. *Towards Abstractions for Distributed Systems* PhD thesis, Imperial College, Department of Computing, 2002.

[8] M. Bugliesi, G. Castagna, S. Crafa. Boxed Ambients. *Lecture Notes in Computer Science*, vol.2215, 38-63, 2001.

[9] C.S. Calude, Gh. Păun. Bio-Steps Beyond Turing. *Biosystems*, vol.77(1-3), 175–194, 2004.

[10] L. Cardelli, A. Gordon. Mobile Ambients. *Theoretical Computer Science*, vol.240(1), 170-213, 2000.

[11] L. Cardelli. Brane Calculi - Interactions of Biological Membranes. *Lecture Notes in Computer Science*, vol.3082, 257–280, 2005.

[12] L. Cardelli, Gh. Păun. An Universality Result for a (Mem)Brane Calculus Based on Mate/Drip Operations. *International Journal of Foundations of Computer Science*, vol.17(1), 49–68, 2006.

[13] M. Cavaliere, V. Deufemia. Further Results on Time-Free P Systems. *International Journal on Foundational Computer Science*, vol.17(1), 69–89, 2006.

[14] M. Cavaliere, I. Mura. Experiments on the Reliability of Stochastic Spiking Neural P Systems. *Natural Computing*, vol.7(4), 453–470, 2008.

[15] M. Cavaliere, D. Sburlan. Time-Independent P Systems. *Lecture Notes in Computer Science*, vol.3365, 239–258, 2005.

[16] M. Cavaliere, D. Sburlan. Time and Synchronization in Membrane Systems. *Fundamenta Informaticae*, vol.64(1-4), 65–77, 2005.

[17] M. Cavaliere, R. Freund, A.Leitsch, Gh. Păun. Event-Related Outputs of Computations in P Systems. *Journal of Automata, Languages and Combinatorics*, vol.11(3), 263–278, 2006.

[18] M. Cavaliere, C. Zandron. Time-Driven Computations in P Systems. *Proceedings of Fourth Brainstorming Week on Membrane Computing*, 133–143, 2006.

[19] G. Ciobanu, B. Aman. On the Relationship Between Membranes and Ambients. *Biosystems*, vol.91(3), 515–530, 2008.

[20] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (editors). *Applications of Membrane Computing*, Springer, Natural Computing Series, 2006.

[21] G. Ciobanu, C. Prisacariu. Timers for Distributed Systems. *Electronic Notes in Theoretical Computer Science*, vol.164(3), 81–99, 2006.

[22] G. Ciobanu, C. Prisacariu. Coordination by Timers for Channel-Based Anonymous Communications. *Electronic Notes in Theoretical Computer Science*, vol.175(2), 3–17, 2007.

[23] D. Hirschkoff, D. Teller, P. Zimmer. Using Ambients to Control Resources. *Lecture Notes in Computer Science*, vol.2421, 288-303, 2002.

[24] O.H. Ibarra, A. Păun. Computing Time in Computing with Cells. *Lecture Notes In Computer Science*, vol.3892, 112–128, 2006.

[25] C.A. Janeway, P. Travers, M. Walport, M.J. Shlomchik. *Immunobiology - The Immune System in Health and Disease*. Fifth Edition. Garland Publishing, 2001.

[26] S.N. Krishna, Gh. Păun. P Systems with Mobile Membranes. *Natural Computing*, vol.4(3), 255–274, 2005.

[27] F. Levi, D. Sangiorgi. Controlling Interference in Ambients. *Principles of Programming Languages*, 352-364, 2000.

[28] H. Lodish, A. Berk, P. Matsudaira, C. Kaiser, M. Krieger, M. Scott, L. Zipursky, J. Darnell. *Molecular Cell Biology*. Fifth Edition, 2003.

[29] R. Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[30] D. Molteni, C. Ferretti, G. Mauri. Frequency Membrane Systems. *Computing and Informatics*, vol.27(3), 467–479, 2008.

[31] H. Nagda, A. Păun, A. Rodríguez-Patón. P Systems with Symport/Antiport and Time. *Lecture Notes In Computer Science*, vol.4361, 463–476, 2006.

[32] A. Păun, A. Rodríguez-Patón. On Flip-Flop Membrane Systems with Proteins. *Lecture Notes In Computer Science*, vol.4860, 414–427, 2007.

[33] Gh. Păun. *Membrane Computing. An Introduction*. Springer, 2002.

[34] I. Petre, L. Petre. Mobile Ambients and P Systems. *Journal of Universal Computer Science*, vol.5(9), 588–598, 1999.

[35] Web page of the P systems: `http://ppage.psystems.eu`.

[36] Web page http://bcs.whfreeman.com/thelifewire.

On the Interpretation of Delays in Delay Stochastic Simulation of Biological Systems

Roberto Barbuti Giulio Caravagna Andrea Maggiolo-Schettini
Paolo Milazzo
Dipartimento di Informatica, Università di Pisa
Largo Pontecorvo 3, 56127 Pisa, Italy.
{barbuti,caravagn,maggiolo,milazzo}@di.unipi.it

Delays in biological systems may be used to model events for which the underlying dynamics cannot be precisely observed. Mathematical modeling of biological systems with delays is usually based on Delay Differential Equations (DDEs), a kind of differential equations in which the derivative of the unknown function at a certain time is given in terms of the values of the function at previous times. In the literature, delay stochastic simulation algorithms have been proposed. These algorithms follow a "delay as duration" approach, namely they are based on an interpretation of a delay as the elapsing time between the start and the termination of a chemical reaction. This interpretation is not suitable for some classes of biological systems in which species involved in a delayed interaction can be involved at the same time in other interactions. We show on a DDE model of tumor growth that the delay as duration approach for stochastic simulation is not precise, and we propose a simulation algorithm based on a "purely delayed" interpretation of delays which provides better results on the considered model.

1 Introduction

Biological systems can often be modeled at different abstraction levels. A simple event in a model that describes the system at a certain level of detail may correspond to a rather complex network of events in a lower level description. The choice of the abstraction level of a model usually depends on the knowledge of the system and on the efficiency of the analysis tools to be applied to the model.

Delays may appear in models of biological systems at any abstraction level, and are associated with events whose underlying dynamics either cannot be precisely observed or is too complex to be handled efficiently by analysis tools. Roughly, a delay may represent the time necessary for the underlying network of events to produce some result observable in the higher level model.

Mathematical modelling of biological systems with delays is mainly based on delay differential equations (DDEs), a kind of differential equations in which the derivative of the unknown function at a certain time is given in terms of the values of the function at previous times. In particular, this framework is very general and allows both simple (constant) and complex (variable or distributed) forms of delays to be modeled.

As examples of DDE models of biological systems we mention [3, 15, 10, 14, 7]. In [3, 15] an epidemiological model is defined that computes the theoretical number of people infected with a contagious illness in a closed population over time; in the model a delay is used to model the length of the infectious period. In [10] a simple predator-prey model with harvesting and time delays is presented; in the model a constant delay is used based on the assumption that the change rate of predators depends on the number of prey and predators at some previous time. Finally, models of tumor growth [14] and of HIV cellular infection [7] have been presented and analyzed by using DDEs.

R.J.Back, I.Petre, E. de Vink (Eds.): Computational
Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 17–29, doi:10.4204/EPTCS.6.2

Models based on DDEs, as their simplest versions based on ordinary differential equations (ODEs), may be studied either analytically (by finding the solution of the equations, equilibria and bifurcation points) or via approximated numerical solutions. However, for complex real models analytical solutions are often difficult or impossible to be computed, whereas their approximated numerical solution is more feasible.

Models based on differential equations, although very useful when dealing with biological systems involving a huge number of components, are not suitable to model systems in which the quantity of some species is small. This is caused by the fact that differential equations represent discrete quantities with continuous variables, and when quantities are close to zero this becomes a too imprecise approximation. In these cases a more precise description of systems behaviour can be obtained with stochastic models, where quantities are discrete and stochastic occurrence of events is taken into account.

The most common analysis technique for stochastic models is stochastic simulation that, in the case of models of biological systems without delays, often exploits Gillespie's Stochastic Simulation Algorithm (SSA) of chemical reactions [9], or one of its approximated variants [8, 6]. In recent years, the interest for stochastic delayed processes increased [13]. In [2] a Delay Stochastic Simulation Algorithm (DSSA) has been proposed, this algorithm gives an interpretation as durations to delays. The delay associated with a chemical reaction whose reactants are consumed (i.e. are not also products) is interpreted as the duration of the reaction itself. Such an interpretation implies that the products of a chemical reaction with a delay are added to the state of the simulation not at the same time of reactants removal, but after a quantity of time corresponding to the delay. Hence, reactants cannot be involved in other reactions during the time modeled by the delay.

We argue that the interpretation of delay as duration is not always suitable for biological systems. We propose a simple variant of the DSSA in which reactants removal and products insertion are performed together after the delay. This corresponds to a different interpretation of delays, that is the delay is seen as the time needed for preparing an event which happens at the end of the delay. An example of a biological behavior which can be suitably modelled by this interpretation is mitosis. Cell mitosis is characterized by a pre–mitotic phase and by a mitotic phase (cell division). The pre–mitotic phase prepares the division of the cell, when a cell undergoes the mitotic process, the pre–mitotic phase can be seen as a delay before the real cell division. During the pre–mitotic phase the cell can continue to interact with the environment, for example it can die. The DSSA in [2] cannot model this interactions because the reactants (in this case the cell itself) are removed at the beginning of reaction and the products are added at its end (that is after the delay).

In this paper we start by recalling the definition of DDEs and a DDE model of tumor growth [14]. Then, we give a stochastic model of the considered tumor growth example and simulate it by using the DSSA introduced in [2] and based on an interpretation of delays as durations. Finally, we propose a new interpretation of delays and, consequently, a new variant of the DSSA that we apply to the considered tumor growth example. At the end of the paper we discuss further improvements of our approach and we draw some conclusions.

2 Delay Differential Equations (DDEs)

The mathematical modeling of biological systems is often based on Ordinary Differential Equations (ODEs) describing the dynamics of the considered systems in terms of variation of the quantities of the involved species over time.

Whenever phenomena presenting a delayed effect are described by differential equations, we move

from ODEs to *Delay Differential Equations* (DDEs). In DDEs the derivatives at current time depend on some past states of the system. The general form of a DDE for $X(t) \in \mathbb{R}^n$ is

$$\frac{dX}{dt} = f_x(t, X(t), X_t),$$

where $X_t = \{X(t') : t' \leq t\}$ represents the trajectory of the solution in the past.

The simplest form of DDE considers *constant delays*, namely consists of equations of the form

$$\frac{dX}{dt} = f_x(t, X(t), X(t-\sigma_1), \ldots, X(t-\sigma_n))$$

with $\sigma_1 > \ldots > \sigma_n \geq 0$ and $\sigma_i \in \mathbb{R}$. This form of DDE allows models to describe events having a fixed duration. They have been used to describe biological systems in which events have a non-negligible duration [3, 15, 10] or in which a sequence of simple events is abstracted as a single complex event associated with a duration [14, 7].

In what follows we recall an example of DDE model of a biological system that we shall use to compare delay stochastic simulation approaches.

2.1 A DDE model of tumor growth

Villasana and Radunskaya proposed in [14] a DDE model of tumor growth that includes the immune system response and a phase-specific drug able to alter the natural course of action of the cell cycle of the tumor cells.

The cell cycle is a series of sequential events leading to cell replication via cell division. It consists of four phases: G_1, S, G_2 and M. The first three phases (G_1, S, G_2) are called interphase. In these phases, the main event which happens is the replication of DNA. In the last phase (M), called mitosis, the cell segregates the duplicated sets of chromosomes between daughter cells and then divides. The duration of the cell cycle depends on the type of cell (e.g a human normal cell takes approximately 24 hours to perform a cycle).

The model in [14] considers three populations of cells: the immune system, the population of tumor cells during cell cycle interphase, and the population of tumor cells during mitosis. A delay is used to model the duration of the interphase, hence the model includes a delayed event that is the passage of a tumor cell from the population of those in the interphase to the population of those in the mitotic phase. In the model the effect of a phase-specific drug, able to arrest tumor cells during the mitosis, is studied. Such a drug has a negative influence also on the survival of cells of the immune system.

In this paper we study a simplified version of the model (presented in subsection 4.1.2 of [14]), where the effects of the immune response and of the drug are not taken into account. The simplified model, which considers only tumor cells (both in pre-mitotic and mitotic phases), consists of the following DDEs:

$$\frac{dT_I}{dt} = 2a_4 T_M - d_2 T_I - a_1 T_I(t-\sigma) \qquad T_I(t) = \phi_0(t) \text{ for } t \in [-\sigma, 0]$$

$$\frac{dT_M}{dt} = a_1 T_I(t-\sigma) - d_3 T_M - a_4 T_M \qquad T_M(t) = \phi_1(t) \text{ for } t \in [-\sigma, 0]$$

Function $T_I(t)$ denotes the population of tumor cells during interphase at time t, and function $T_M(t)$ denotes the tumor population during mitosis at time t. The terms $d_2 T_I$ and $d_3 T_M$ represent cell deaths, or apoptoses. The constants a_1 and a_4 represent the phase change rates from interphase to mitosis (a_1)

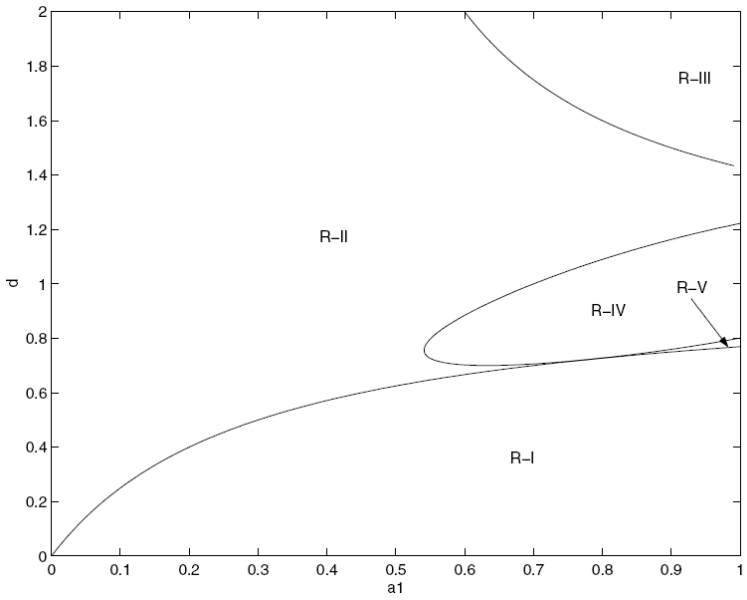


Figure 1: The regions which describe the different behaviours of the DDE model by varying parameters a_1 and d (picture taken from [14]).

and back (a_4). In the following we shall denote with d the rate at which mitotic cells disappear, namely $d = d_3 + a_4$.

We assume that cells reside in the interphase at least σ units of time; then the number of cells that enter mitosis at time t depends on the number of cells that entered the interphase σ units of time before. This is modeled by the terms $T_I(t-\sigma)$ in the DDEs. Note that each cell leaving the mitotic phase produces two new cells in the T_I population (term $2a_4T_M$). In the model the growth of the tumor cell population is obtained only through mitosis, and is given by the constants a_1, a_4, and σ which regulate the pace of cell division. The delay σ requires the values of T_I and T_M to be given also in the interval $[-\sigma,0]$: such values are assumed to be constant in the considered interval, and hence equal to the values of T_I and T_M at time 0.

The analytic study of the DDEs constituting the model gives $(0,0)$ as unique equilibrium. In Figure 1 (taken from [14]) some results are shown of the study of the model by varying a_1, d and σ and by setting the parameters a_4 and d_2 to 0.5 and 0.3, respectively. Figure 1 shows five regions.

When $\sigma = 0$, the region in which the tumor grows is R-I, while in the other regions the tumor decays.

When the delay is present ($\sigma > 0$), the growth region is essentially unaltered, but the decay is split in regions in which the tumor has different behaviours: in regions R-II $\cup$ R-IV the tumor still decays, but in regions R-III $\cup$ R-V, when the value of σ is sufficiently large, the equilibrium becomes unstable. This is shown in Figures 2 and 3.

Figure 2 describes the behaviour of the model, obtained by numerical solutions, inside the regions R-I, R-II, R-III, and R-IV, when $\sigma = 1$. Actually, we considered the point $(0.6,0.6)$ in R-I, the point $(0.4,1.0)$ in R-II, the point $(1.0,1.8)$ in R-III, the point $(0.8,0.8)$ in R-IV and an initial state consisting in 10^5 tumor cells in the interphase and 10^5 tumor cells in mitosis. We shall use always this parameters in the rest of the paper. In the figure, we can observe that, while the tumor grows in region R-I, it decays in all the others.

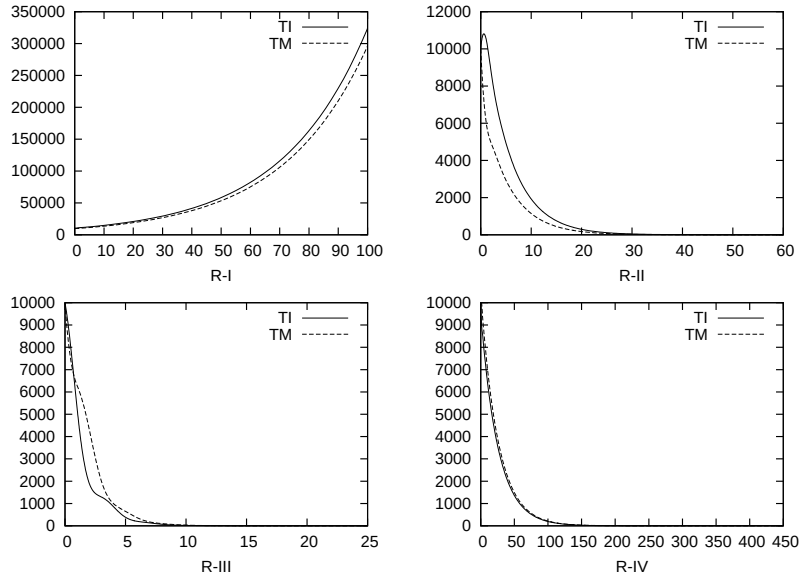


Figure 2: Results of the numerical solution of the DDE model with $\sigma = 1$ for the regions described in Figure 1. On the x-axis time is given in *days* and on the y-axis is given the *number of cells*.

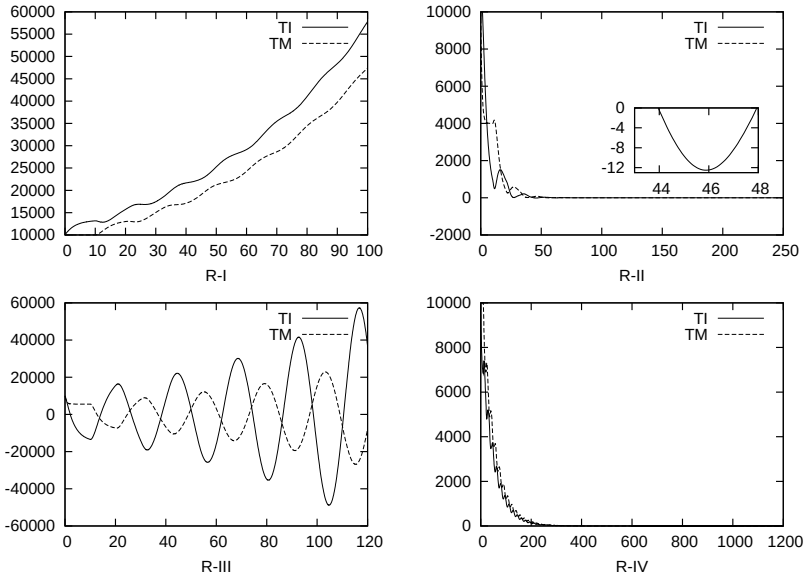


Figure 3: Results of the approximated numerical simulation of the DDE model with $\sigma = 10$ for the regions described in Figure 1. On the x-axis time is given in *days* and on the y-axis is given the *number of cells*.

Figure 3 describes the behaviour of the model when $\sigma = 10$. In regions R-I and R-IV the tumor has the same behaviour as before. In region R-II it decays after some oscillations, while in region R-III it expresses an instability around the equilibrium. However, remark that values of T_M and T_I under 0 are not realistic, and, as we will see in the following, they cannot be obtained by stochastic simulations.

3 Delay Stochastic Simulation

In this section we present algorithms for the stochastic simulation of biological systems with delays. Firstly, we introduce a well–known formulation of one of these algorithms and we analyze the results of the simulations of the stochastic model equivalent to the one presented in the previous section. Secondly, we propose a variant of this algorithm and we compare the results of the simulations done by using this algorithm with those of the simulation done by using the original one.

All the simulations and the algorithms that we are going to present in this section have been implemented in the software tool `Delay Sim`. This tool, avalaible at `http://www.di.unipi.it/msvbio`, has been written in Java.

3.1 The Delay as Duration Approach (DDA)

In [2] Barrio *et al.* introduced a *Delay Stochastic Simulation Algorithm* (DSSA) by adding delays to Gillespie's Stochastic Simulation Algorithm (SSA) [9]. The algorithm has been used to explain more carefully than with DDE models the observed sustained oscillations in the expression levels of some proteins.

In order to recall the definition of the algorithm in [2] we consider a well–stirred system of *molecules* of N chemical *species* $\{S_1,\ldots,S_N\}$ interacting through M chemical *reaction channels* $\mathscr{R} = R_1,\ldots,R_M$. We assume the volume and the temperature of the system to be constant. We denote the number of molecules of species S_i in the system at time t with $X_i(t)$, and we want to study the evolution of the *state vector* $\mathbf{X}(t) = (X_1(t),\ldots,X_N(t))$, assuming that the system was initially in some state $\mathbf{X}(t_0) = \mathbf{x}_0$.

A reaction channel R_j is characterized mathematically by three quantities. The first is its *state–change vector* $\nu_j = (\nu_{1j},\ldots,\nu_{Nj})$, where ν_{ij} is defined to be the change in the S_i molecular population caused by one R_j reaction; let us denote each state–change vector ν_j as a the composition of the state–change vector for reactants, ν_j^r, and the state–change vector for products, ν_j^p, noting that $\nu_j = \nu_j^r + \nu_j^p$. For instance, given two species A and B, a reaction of the form $A \rightarrow B$ is described by the vector of reactants $(-1,0)$, by the vector of products $(0,1)$ and by the state–change vector $(-1,1)$; differently, a reaction of the form $A \rightarrow A+B$ is described by the vector of reactants $(-1,0)$, by the vector of products $(1,1)$, and by the state–change vector $(0,1)$.

The second characterizing quantity for a reaction channel R_j is its *propensity function* $a_j(\mathbf{x})$; this is defined, accordingly to [9], so that, given $\mathbf{X}(t) = \mathbf{x}$, $a_j(\mathbf{x})dt$ is the probability of reaction R_j to occur in state $\mathbf{x}$ in the time interval $[t, t+dt]$. As stated in [9], the probabilistic definition of the propensity function finds its justification in physical theory.

The other characterizing quantity is a constant delay defined by a real number $\sigma \geq 0$. Following Barrio *et al.*, we classify reactions with delays into two categories: non-consuming reactions, where the reactants are also products (e.g. $A \rightarrow A+B$), and consuming reactions, where some of the reactants are consumed (e.g. $A \rightarrow B$). Throughout the paper, we denote the set of non-consuming reactions with delay by $\mathscr{R}_{nc}$, the set of consuming reactions with delay by $\mathscr{R}_c$, and the reactions without delays by $\mathscr{R}_{nd}$; notice that $\mathscr{R} = \mathscr{R}_{nc} \cup \mathscr{R}_c \cup \mathscr{R}_{nd}$.

Algorithm **DSSA with "delays as duration approach"**

1. Initialize the time $t = t_0$ and the system state $\mathbf{x} = \mathbf{x}_0$.
2. Evaluate all the $a_j(\mathbf{x})$ and their sum $a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})$;
3. Given two random numbers r_1, r_2 uniformly distributed in the interval $[0;1]$, generate values for τ and j in accordance to

$$\tau = \frac{1}{a_0(\mathbf{x})} \ln(\frac{1}{r_1}) \qquad \sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x})$$

   (A) If delayed reaction R_k is scheduled at time $t + \tau_k$ and $\tau_k < \tau$
      (A1) If $R_k \in \mathscr{R}_{nc}$ then update $\mathbf{x} = \mathbf{x} + \nu_k$ and $t = t + \tau_k$;
      (A2) If $R_k \in \mathscr{R}_c$ then update $\mathbf{x} = \mathbf{x} + \nu_k^p$ and $t = t + \tau_k$;
   (B) else:
      (B1) If $R_j \in \mathscr{R}_{nd}$ then update $\mathbf{x} = \mathbf{x} + \nu_j$ and $t = t + \tau$;
      (B2) If $R_j \in \mathscr{R}_{nc}$, schedule R_j at time $t + \sigma_j + \tau$ and set time to $t + \tau$;
      (B3) If $R_j \in \mathscr{R}_c$, schedule R_j at time $t + \sigma_j + \tau$, update $\mathbf{x} = \mathbf{x} + \nu_k^r$ and set time to $t + \tau$;
4. go to step 2.

Figure 4: The DSSA with "delays as duration approach" proposed in [2].

By adding delays to the SSA, Barrio *et al.* provide a method to model the firing of a reaction with delay based on the previously given classification. Formally, given a system in state $\mathbf{X}(t) = \mathbf{x}$, let us denote with τ the stochastic time quantity computed as in the SSA representing the putative time for next reaction to fire. Let us assume to choose to fire a non-consuming reaction with delay (a reaction from set $\mathscr{R}_{nc}$); then the reaction is scheduled at time $t + \sigma + \tau$ where σ is the delay of the reaction. Furthermore, the clock is increased to the value $t + \tau$ and the state does not change. On the contrary, if a consuming reaction with delay (a reaction from set $\mathscr{R}_c$) is chosen to fire, then its reactants are immediately removed from the state $\mathbf{x}$, the insertion of the products is scheduled at time $t + \sigma + \tau$, and, finally, the clock is increased to the value $t + \tau$. Reactions from set $\mathscr{R}_{nd}$ (non–delayed reactions) are dealt with exactly as in the SSA. The DSSA by Barrio *et al.* is given in Figure 4.

We discuss now on the scheduling of the reactions with delay. When a non-consuming reaction is chosen, the algorithm does not change state, but simply schedules the firing of the reaction at time $t + \sigma_j + \tau$ (step $(B2)$). The reaction will complete its firing (reactants and products will be removed and inserted, respectively) when performing steps (A) and $(A1)$.

Differently, as regards consuming reactions, the removal of the reactants is done at time instant t (step $(B3)$) preceding the time instant of insertion of the products (steps (A) and $(A2)$), namely the time at which the insertion is scheduled, $t + \sigma_j + \tau$. Notice that the removed reactants cannot have other interactions during the time interval $[t, t + \sigma_j + \tau)$.

As the reactants cannot have other interactions in the time quantity passing between the removal of the reactants and the insertion of the products, then this quantity can be seen as a duration needed for the reactants to exclusively complete the reaction. Since the approach of Barrio *at al.* gives this interpretation of delays we shall call it "delays as duration approach"(DDA).

As regards the handling of the scheduled events (step (A) of the algorithm), if in the time interval

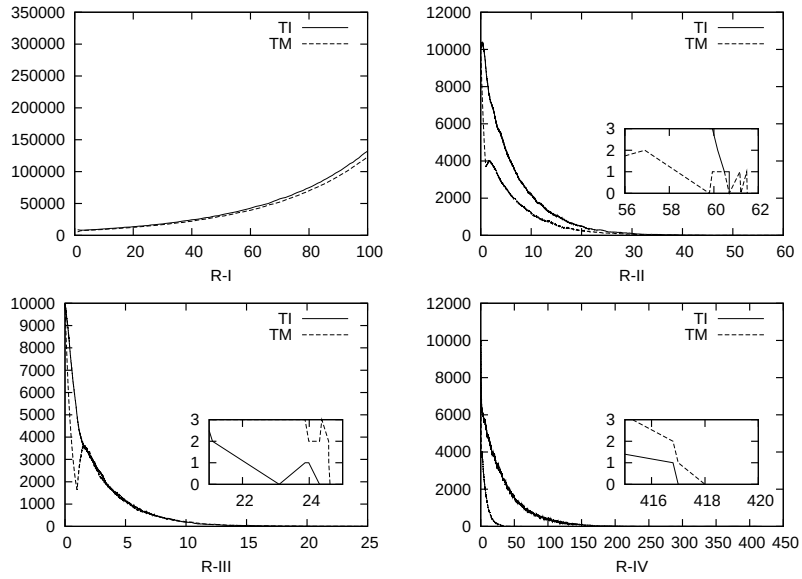


Figure 5: DDA simulation of the stochastic model with $\sigma = 1$ for the regions described in Figure 1. On the x-axis time is given in *days* and on the y-axis is given the *number of cells*.

$[t;t+\tau)$ there are scheduled reactions, then τ is rejected and the scheduled reaction is handled. Since generating random numbers is a costly operation, other authors defined variants of the DSSA that avoid rejecting τ in the handling of scheduled reactions [5, 1]. However, the interpretation of the delays used to define these variants is the same as that of Barrio *et al.*.

This interpretation of delays may not be precise for all biological systems. In particular, it may be not precise if in the biological system the reactants can have other interactions during the time window modeled by the delay. The tumor growth system we have recalled in Section 2.1 is an example of these systems. In fact, while tumor cells are involved in the phase change from interphase to mitosis (the delayed event) they can also die.

We applied the DSSA by Barrio *at al.* (we refer to the simulations done by applying this DSSA as DDA simulations) to a chemical reaction model corresponding to the DDE model of tumor growth recalled in Section 2.1. The reactions of the model are the following:

$$T_I \xrightarrow{a_1} T_M \text{ with delay } \sigma \qquad T_M \xrightarrow{a_4} 2T_I \qquad T_I \xrightarrow{d_2} \qquad T_M \xrightarrow{d_3} .$$

We have run 100 simulations for each considered parameter setting. The results of simulations with the same parameters as those considered in Figures 2 and 3 are shown in Figures 5 and 6, respectively. Actually, in the figures we show the result of one randomly chosen simulation run for each parameter setting.

Qualitatively, results obtained with DDA simulations are the same as those obtained with numerical simulation of the DDEs: we have exponential tumor growth in region R-I, tumor decay in the other regions and oscillations arise when the delay is increased. However, from the quantitative point of view we have that in the DDA simulations the growth in region R-I and the decay in the other regions are always slower than in the corresponding numerical simulation of the DDEs. In fact, with $\sigma = 1$ by the numerical simulation of the DDEs we have that in region R-I after 100 days both the quantities of tumor

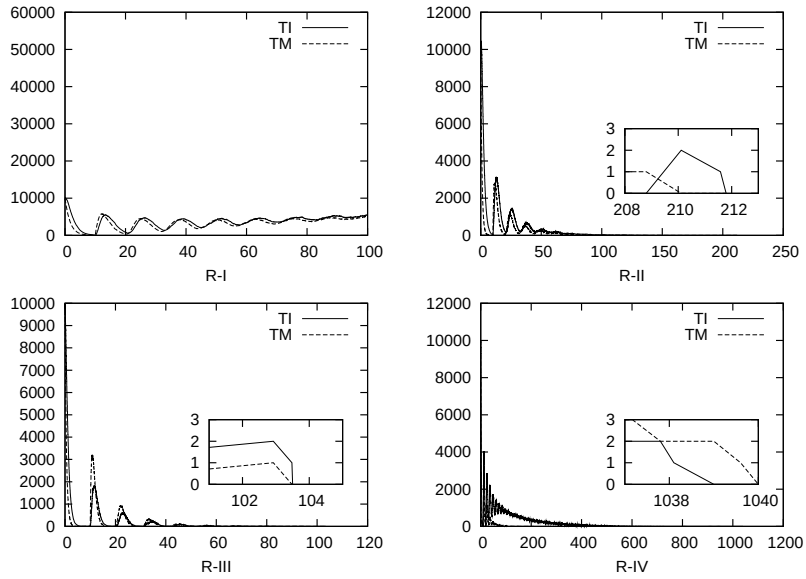


Figure 6: DDA simulation of the stochastic model with $\sigma = 10$ for the regions described in Figure 1. On the x-axis time is given in *days* and on the y-axis is given the *number of cells*.

cells in interphase and in mitotic phase are around 300000, while in the result of DDA simulations they are around 130000. In the same conditions, but with $\sigma = 10$, in the numerical simulation of the DDEs we have about 47000 tumor cells in mitosis and 57000 tumor cells in interphase, while in the DDA simulations we have about 5000 and 5500 cells, respectively. As regards the other regions, in Table 1 the average tumor eradication times obtained with DDA simulations are compared with those obtained with numerical simulation of the DDEs (in this case with "eradication" we mean that the number of tumor cells of both kinds is under the value 1). Again, we have that in DDA simulations the dynamics is slower than in the numerical simulation of the DDEs. For instance, with $\sigma = 10$, in region R-IV the time needed for eradication in the DDEs is about 41% of the time needed in the DDA (440 against 1072), in region R-II the percentage is smaller, 26% (59 against 224), and, in region R-III, it reaches 9% (12 against 126). For the same regions with $\sigma = 1$ these differences are smaller but not negligible.

3.2 A Purely Delayed Approach (PDA)

In this section we propose a variant of the DSSA based on a different interpretation of delays, namely a Stochastic Simulation Algorithm which follows a "purely delayed approach" (PDA). With this interpretation we try to overcome the fact that in the DDA the reactants cannot have other interactions. Furthermore, differently from Barrio *et al.*, we use the same interpretation of delays to define the method for firing both non-consuming and consuming reactions. This interpretation of delays was firstly implicitly adopted by Bratsun *et al.* in [4], to model a very simple example of protein degradation.

The approach we propose consists in firing a reaction completely when its associated scheduled events is handled, namely removing its reactants and inserting its products after the delay. The fact that we simply schedule delayed reactions without immediately removing their reactants motivates the terminology of "purely delayed". Notice that non-consuming reactions are handled in the same way by DDA and PDA.

Algorithm **DSSA with "purely delayed approach"**

1. Initialize the time $t = t_0$ and the system state $\mathbf{x} = \mathbf{x}_0$.
2. Evaluate all the $a_j(\mathbf{x})$ and their sum $a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})$;
3. Given two random numbers r_1, r_2 uniformly distributed in the interval $[0,1]$, generate values for τ and j in accordance to

$$\tau = \frac{1}{a_0(t)} \ln(\frac{1}{r_1}) \qquad \sum_{i=1}^{j-1} a_i(\mathbf{X}(t)) < r_2 \cdot a_0(t) \leq \sum_{i=1}^{j} a_i(\mathbf{X}(t))$$

   (a) If delayed reaction R_k is scheduled at time $t + \tau_k$ and $\tau_k < \tau$ and $\nu_k^r \prec \mathbf{x}$, then update $\mathbf{x} = \mathbf{x} + \nu_k$ and $t = t + \tau_k$;

   (b) else, schedule R_j at time $t + \sigma_j + \tau$, set time to $t + \tau$;

4. go to step 2.

Figure 7: The DSSA with "purely delayed approach".

	DDEs	DDA Simulation	PDA Simulation
R-II with $\sigma = 1.0$	50	64	51
R-II with $\sigma = 10.0$	59	224	67
R-III with $\sigma = 1.0$	15	29	17
R-III with $\sigma = 10.0$	12	126	20
R-IV with $\sigma = 1.0$	238	302	214
R-IV with $\sigma = 10.0$	440	1072	248

Table 1: Average eradication times given in *days* for DDE model, DDA and PDA stochastic models. For the stochastic models the entries represent the sample of 100 simulations.

In this interpretation of delays it may happen that, when handling a scheduled reaction, the reactants may not be present in the current state. In fact, they could have been destroyed or transformed by other interactions happened after the scheduling. In this case, the scheduled reaction has to be ignored. To formalize this, we know that a reaction R_j can be applied only if its reactants are all present in the current state of the simulation. Algebraically this corresponds to the fact that $\nu_j^r \prec \mathbf{x}$ where ν_j^r is the state–change vector of the reactants of reaction R_j, the system is described by $\mathbf{x}$ and $\prec$ is the ordering relation defined as $\forall i = 1, \ldots, N. \ -\nu_{ij}^R \leq X_i(t)$. In order to verify that a scheduled reaction can effectively fire, it will be sufficient to check whether this condition holds. The formal definition of the DSSA with PDA is given in Figure 7.

As for the DDA, we have run 100 simulations of the stochastic model of tumor growth for each considered parameter setting. The results of simulations (we refer to these simulations as PDA simulations) with the same parameters as those considered in Figures 2 and 3 are shown in Figures 8 and 9, respectively. Actually, in the figures we show the result of one randomly chosen simulation run for each parameter setting.

Qualitatively, results obtained with PDA simulations are the same as those obtained with numerical simulation of the DDEs (and with DDA simulations). From the quantitative point of view we have that in

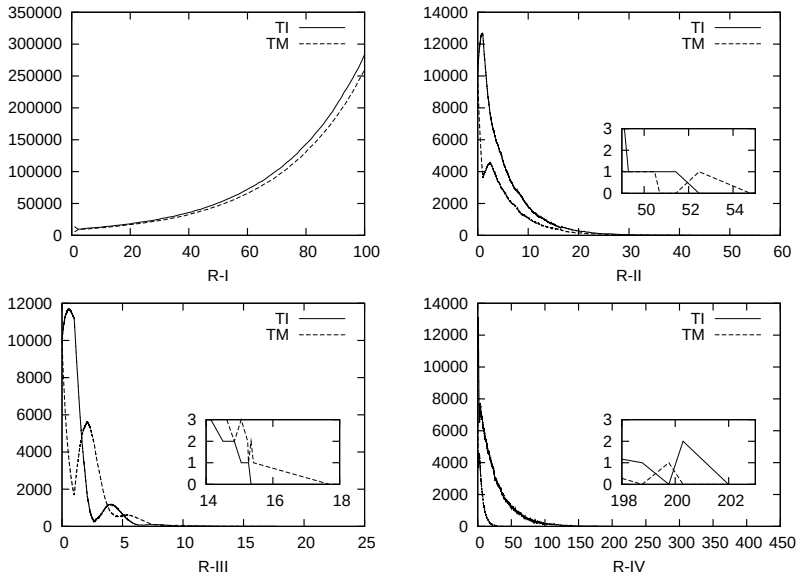


Figure 8: PDA simulation of the stochastic model with $\sigma = 1$ for the regions described in Figure 1. On the x-axis time is given in *days* and on the y-axis is given the *number of cells*.

the PDA simulations the growth in region R-I with $\sigma = 1$ is almost equal to the corresponding numerical simulation of the DDEs (about 300000 tumor cells in both mitosis and interphase after 100 days, we recall that the DDA had reached values around 130000). On the contrary, with $\sigma = 10$, the difference between DDEs and PDA is higher: we have about 22000 tumor cells in interphase against 57000 for the DDEs and 5500 for the DDA, and 16000 tumor cells in mitosis against 47000 for the DDEs and 5000 for the DDA.

As regards the other regions, in Table 1 the average tumor eradication times obtained with PDA simulations are compared with those obtained with numerical simulation of the DDEs (again, in this case with "eradication" we mean that the number of tumor cells of both kinds is under the value 1). In PDA simulations the dynamics is generally slower than in the numerical simulation of the DDEs but it is faster than the DDA one. With $\sigma = 10$, in region R-IV the time needed for eradication in the PDA is smaller than the one in the DDEs (248 days against 440, DDA is 1072). In region R-II the values are: 67 days for the PDA and 59 days for the DDEs, DDA is 224. In region R-III values are: 20 days for the PDA, 12 days for the DDEs, and 126 days for DDA.

It is important to remark that differences between delay stochastic simulation results and numerical solutions of DDEs are also influenced by the initial conditions. The numerical solution of the DDEs assumes the initial population to be constant and greater than zero in the time interval $[-\sigma, 0]$. This allows delayed event to be enabled in the time interval $[0, \sigma]$. Both variants of the DSSA start to schedule delayed events from time 0, hence delayed reactions can fire only after the time σ. This result, when σ is great enough, in a behaviour that is, in general, delayed with respect to that given by the DDEs.

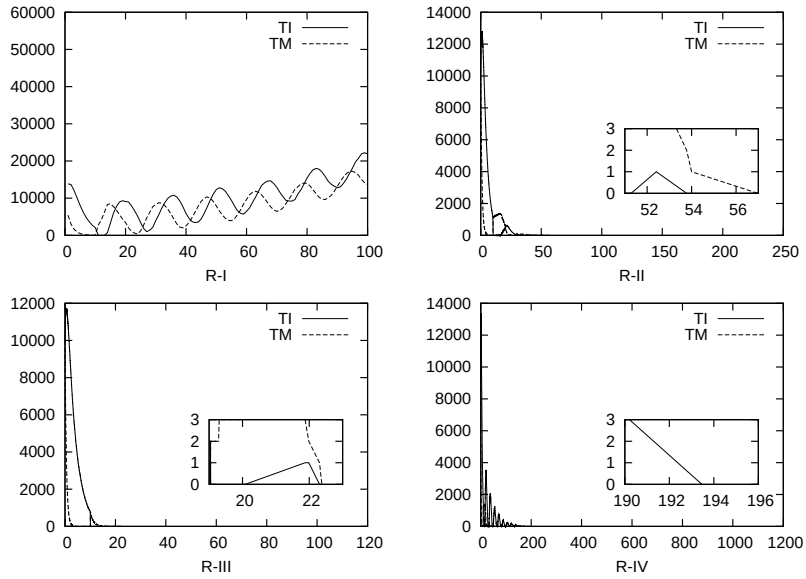


Figure 9: PDA simulation of the stochastic model with $\sigma = 10$ for the regions described in Figure 1. On the x-axis time is given in *days* and on the y-axis is given the *number of cells*.

4 Discussion

In the previous sections we showed two different approaches to the firing of delayed reactions. The two approaches can be conveniently used for dealing with two different classes of delayed reactions. The delay as duration approach suitably deals with reactions in which reactants cannot participate, whenever scheduled, in other reactions. On the other hand, the purely delayed approach can be conveniently used in cases in which reactants can be involved in other reactions during the delay time.

These two different notions of delay have been presented in the framework of Petri nets with time information. In particular, in Timed nets [12] a notion of delay similar to a duration appears; differently, in Time nets [11] the notion of delay corresponds to our purely delayed approach.

In the example we have shown, cells in the interphase, which wait for entering the mitotic phase, can be involved in another reaction, namely their death. Thus in this example the purely delayed approach seems to be more appropriate for capturing the behaviour of the real system. Obviously, there are biological systems in which, due to the heterogeneity of reactions, both the approaches should be used and we plan to investigate, in the future, the possibility of combining the two approaches in a unique framework.

References

[1] D.F. Anderson (2007): *A Modified Next Reaction Method for Simulating Chemical Systems with Time Dependent Propensities and Delays. J. Ch. Phys.* 127(21), 214107.

[2] M. Barrio, K. Burrage, A. Leier, T. Tian (2006) : *Oscillatory Regulation of Hes1: Discrete Stochastic Delay Modelling and Simulation. PLoS Computational Biology*, 2(9).

[3] E. Beretta, T. Hara, W. Ma, Y. Takeuchi (2002): *Permanence of an SIR Epidemic Model with Distributed Time Delays. Tohoku Mathematical Journal 54(2), 581–591.*

[4] D. Bratsun, D. Volfson, L.S. Tsimring, J. Hasty (2005): *Delay-induced Stochastic Oscillations in Gene Regulation*. *PNAS 102(41), 14593–14598.*

[5] X. Cai (2007): *Exact Stochastic Simulation of Coupled Chemical Reactions with Delays*. *J. Ch. Phys., 126, 124108.*

[6] Y. Cao, D. Gillespie, L. Petzold (2005): *The Slow-scale Stochastic Simulation Algorithm*. *J. Ch. Phys. 122, 014116.*

[7] R.V. Culshaw, S. Ruan (2000) : *A Delay–differential Equation Model of HIV Infection of CD4+ T–cells*. *Mathematical Biosciences 165, 27–39.*

[8] D. Gillespie (2001): *Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems.* *J. Phys. Ch. 115, 1716.*

[9] D. Gillespie: *Exact Stochastic Simulation of Coupled Chemical Reactions*. *J. Phys. Ch. 81, 2340.*

[10] A. Martin, S. Ruan (2001): *Predator-prey Models with Delay and Prey Harvesting*. *J. Math. Biol. 43(3), 247–267.*

[11] L. Popova–Zeugmann, M. Heiner, I. Koch (2005): *Time Petri Nets for Modelling and Analysis of Biochemical Networks*. *Fundamenta Informaticae 67, 149–162.*

[12] C. Ramchandani (1974): *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*. *Massachussets Inst. Technol. Res. Rep., MAC-TR 120.*

[13] R. Schlicht, S. Winkler (2008): *A Delay Stochastic Process with Applications in Molecular Biology*. *J. Math. Biol. 57, 613–648.*

[14] M. Villasana, A. Radunskaya (2003): *A Delay Differential Equation Model for Tumor Growth*. *J. Math. Biol. 47, 270–294.*

[15] F. Zhanga, Z. Lia, F. Zhangc (2008): *Global Stability of an SIR Epidemic Model with Constant Infectious Period*. *Applied Mathematics and Computation 199(1), 285–291.*

BioDiVinE: A Framework for Parallel Analysis of Biological Models

J. Barnat[*], L. Brim[†], I. Černá, S. Dražan[‡], J. Fabriková, J. Láník, and D. Šafránek

Faculty of Informatics
Masaryk University
Botanická 68a, Brno, Czech Republic
safranek@fi.muni.cz

H. Ma[§]

Informatics Life-Sciences Institute
School of Informatics, University of Edinburgh
11 Crichton Street, Edinburgh, EH8 9LE, United Kingdom

In this paper a novel tool BioDiVinE for parallel analysis of biological models is presented. The tool allows analysis of biological models specified in terms of a set of chemical reactions. Chemical reactions are transformed into a system of multi-affine differential equations. BioDiVinE employs techniques for finite discrete abstraction of the continuous state space. At that level, parallel analysis algorithms based on model checking are provided. In the paper, the key tool features are described and their application is demonstrated by means of a case study.

1 Introduction

The central interest of systems biology is investigation of the response of the organism to environmental events (extra-cellular or intra-cellular signals). Even in procaryotic organisms, a single environment event causes a response induced by the interaction of several interwoven modules with complex dynamic behaviour, acting on rapidly different time scales. In higher organisms, these modules form large and complex interaction networks. For instance, a human cell contains in the order of 10,000 substances which are involved in 15,000 different types of reactions. This gives rise to a giant interaction network with complex positive and negative regulatory feedback loops.

In order to deal with the complexity of living systems, experimental methods have to be supplemented with mathematical modelling and computer-supported analysis. One of the most critical limitations in applying current approaches to modelling and analysis is their pure scalability. Large models require powerful computational methods, the hardware infrastructure is available (clusters, GRID, multi-core computers), but the parallel (distributed) algorithms for model analysis are still under development.

In this paper we present the tool BioDiVinE for parallel analysis of biological models. BioDiVinE considers the model in terms of chemical equations. The main features of the tool are the following:

- user interface for specification of models in terms of chemical equations
- formal representation of the model by means of multi-affine ODEs
- setting initial conditions and parameters of the kinetics

[*]J. Barnat has been supported by the Czech Science Foundation grant No. 201/09/P497.
[†]L. Brim has been supported by the Grant Agency of Czech Republic grant No. 201/09/1389.
[‡]S. Dražan and J. Fabriková have been supported by the Academy of Sciences of CR grant No. 1ET408050503.
[§]I. Černá, D. Šafránek, and H. Ma have been supported by the FP6 project No. NEST-043235 (EC-MOAN).

R.J.Back, I.Petre, E. de Vink (Eds.): Computational Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 31–45, doi:10.4204/EPTCS.6.3

- setting parameters of the discrete abstraction
- graphical simulation of the discretized state space
- model checking analysis.

As an abstraction method, `BioDiVinE` adapts the rectangular abstraction approach of multi-affine ODEs mathematically introduced in [7] and algorithmically tackled in [17, 5] by means of a parallel on-the-fly state space generator. The character of abstraction provided by this discretization technique is conservative with respect to the most dynamic properties that are of interest. However, there is the natural effect of adding false-positive behaviour, in particular, the abstracted state space includes trajectories which are not legal in the original continuous model.

The structure of the paper is the following. Section 2 gives a brief overview of the underlying abstraction technique and the model checking approach. Section 3 presents in step-by-step fashion the key features of current version of BioDiVinE. Section 4 provides a case study of employing BioDiVinE for analysis of a biological pathway responsible for ammonium transport in bacteria Escherichia Coli.

1.1 Related Work

In our previous work [3] we have dealt with parallel model checking analysis of piece-wise affine ODE models [14]. The method allows fully qualitative analysis, since the piece-wise affine approximation of the state space does not require numerical enumeration of the equations. Therefore that approach, in contrast to the presented one, is primarily devoted for models with unknown kinetic parameters. The price for this feature is higher time complexity of the state space generation. In particular, time appears there more critical than space while causing the parallel algorithms not to scale well.

In the current version of BioDiVinE all the kinetic parameters are required to be numerically specified. In such a situation there is an alternative possibility to do LTL model checking directly on numerical simulations [18, 8]. However, in the case of unknown initial conditions there appears the need to provide large-scale parameter scans resulting in huge number of simulations. On the contrary, the analysis conducted with BioDiVinE can be naturally generalised to arbitrary intervals of initial conditions by means of rectangular abstraction.

Rectangular abstraction of dynamic systems has been employed in [17] for reachability analysis. The method has been supported by experiments performed on a sequential implementation in Matlab. The provided experiments have showed that for models with 10 variables the reachability analysis ran out of memory after 2 hours of computation. This gave us the motivation to employ parallel algorithms. Moreover, we generalise the analysis method to full LTL model checking.

There is another work that employs rectangular abstraction for dynamic systems [6]. The framework is suitable for deterministic modelling of genetic regulatory networks. The rectangular abstraction relies on replacing S-shaped regulatory functions with piece-wise linear ramp functions. The partitioning of the state space is determined by parameters of the ramp functions. In contrast to that work, we consider directly general multi-affine systems with arbitrarily defined abstraction partitions.

2 Background

2.1 Modelling Approach

The most widely used approach to modelling a system of biochemical reactions is the continuous approach of ordinary differential equations (ODEs). We consider a special class of ODE systems in the

form $\dot{x} = f(x)$ where $x = (x_1, \ldots, x_n)$ is a vector of variables and $f = (f_1, ..., f_n) : \mathbb{R}^n \to \mathbb{R}^n$ is a vector of multiaffine functions. A multiaffine function is a polynomial in the variables $x_1, ..., x_n$ where the degree of any variable is at most 1. Variables x_i represent continuous concentrations of species. Multiaffine ODEs can express reactions in which the stoichiometry coefficients of all reactants are at most 1. The system of ODEs can be constructed directly from the stoichiometric matrix of the biochemical system [12].

$$A + B \xrightarrow{k} C \qquad \begin{aligned} \tfrac{dA}{dt} &= -k \cdot A \cdot B \\ \tfrac{dB}{dt} &= -k \cdot A \cdot B \\ \tfrac{dC}{dt} &= k \cdot A \cdot B \end{aligned}$$

Figure 1: Example of a multi-affine system

Multi-affine system is achieved from the system of biochemical reactions by employing the law of mass action. In Figure 1 there is an example of a simple biochemical system represented mathematically as a multi-affine system. If all the reactions are of the first-order, in particular, the number of reactants in each reaction is at most one, then the system falls into a specific subclass of dynamic systems – the resulting ODE system is affine. An example of an affine system is given in Figure 2.

$$\begin{aligned} A &\xrightarrow{k_1} B + C \\ B &\underset{k_3}{\overset{k_2}{\leftrightarrows}} C \end{aligned} \qquad \begin{aligned} \tfrac{dA}{dt} &= -k_1 \cdot A \\ \tfrac{dB}{dt} &= k_1 \cdot A - k_3 \cdot B + k_2 \cdot C \\ \tfrac{dC}{dt} &= k_1 \cdot A - k_2 \cdot C + k_3 \cdot B \end{aligned}$$

Figure 2: Example of an affine system

2.2 Abstraction Procedure

The rectangular abstraction method employed in BioDiVinE relies on results of de Jong, Gouzé [15] and Belta, Habets [7]. Each variable is assigned a set of specific (arbitrarily defined) points, so-called *thresholds*, expressing concentration levels of special interest. This set contains two specific thresholds – the maximal and the minimal concentration level. Existence of these two thresholds comes from the biophysical fact that in any living organism each biochemical species cannot unboundedly increase its concentration. The intermediate thresholds then define a partition of the (bounded) continuous state space. The individual regions of the partition are called *rectangles*. An example of a partition is given in Figure 3.

The partition of the system gives us directly the finite discrete abstraction of the dynamic system. In particular, BioDiVinE implements a (discrete) state space generator that constructs a finite automaton representing the rectangular abstraction of the system dynamics. Since the states of the automaton are made directly by the rectangles, the automaton is called *rectangular abstraction transition system* (RATS). The algorithm for the state space generator of RATS has been presented in [2]. The idea behind this algorithm is based on the results [15, 7]. The main point is that for each rectangle the exit faces are determined. The intuition is depicted in Figure 4. There is a transition from a rectangle to its neighbouring rectangle only if in the vector field considered in the shared face there is at least one vector whose particular component agrees with the direction of the transition. The important result is that in a

$A \overset{k_1}{\underset{k_2}{\leftrightarrows}} B$

$\frac{dA}{dt} = -k_1 \cdot A + k_2 \cdot B$
$\frac{dB}{dt} = k_1 \cdot A - k_2 \cdot B$

thresholds on A: $0, 5, 6, 10$
thresholds on B: $0, 2, 3, 5$

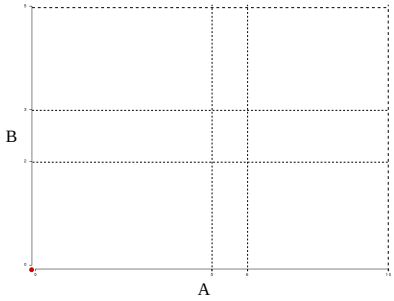


Figure 3: Example of a rectangular partition of a two-dimensional system

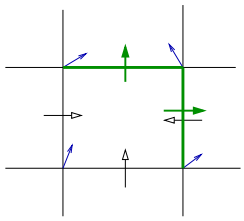


Figure 4: Intuition behind the construction of the rectangular abstraction transition system

multi-affine system it suffices to consider only the vector field in the vertices of the face. In Figure 4, the exit faces of the central rectangle are emphasised by bold lines. In Figure 5 there is depicted the rectangular abstraction transition system constructed for the affine system from Figure 3. It is known that the rectangular abstraction is an overapproximation with respect to trajectories of the original dynamic system.

There is one specific issue when considering the time progress of the abstracted trajectories. If there exists a point in a rectangle from which there is no trajectory diverging out through some exit face, then there is a self-transition defined for the rectangle. In particular, this situation signifies an equilibrium inside the rectangle. Such a rectangle is called non-transient. For affine systems there is known a sufficient and necessary condition that characterises non-transient rectangles by the vector field in the vertices of the rectangle. However, for the case of multi-affine system, only the necessary condition is known. Hence, for multi-affine systems BioDiVinE can treat as non-transient some states which are not necessarily non-transient.

2.3 Model Checking

In the field of formal verification of software and hardware systems, model checking refers to the problem of automatically testing whether a simplified model of a system (a finite state automaton) meets a given specification. Specification is stated by means of a temporal logic formulae. In the setting of RATS, model checking can be used in two basic ways:

1. to automatically detect presence of particular dynamics phenomena in the system
2. to verify correctness of the model (i.e., checking whether some undesired property is exactly avoided)

In the case of dynamic systems we use Linear Temporal Logic (LTL) (see [10] for details on LTL syntax and semantics interpreted on automata). LTL can be also directly interpreted on trajectories

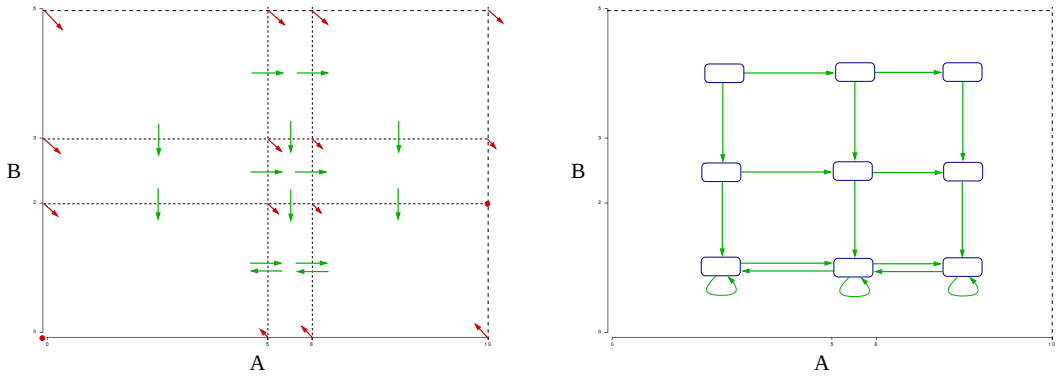


Figure 5: Example of a rectangular abstraction transition system

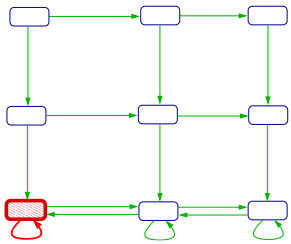


Figure 6: A counterexample contradicting the property of reaching $B > 3$ in finite time

of dynamic systems (see e.g. [13] for definition of the semantics). Given a dynamic system S with a particular initial state we can then say that S satisfies a formula φ, written $S \models \varphi$, only if the trajectory starting at the initial state satisfies φ. In the context of automata, LTL logic is interpreted universally provided that a formula φ is satisfied by the automaton A, written $A \models \varphi$, only if each execution of the automaton starting from any initial state satisfies φ. The following theorem characterises the relation between validity of φ in the rectangular abstraction automaton and in the original dynamic system.

Theorem 2.1 *Consider a dynamic system S and the associated RATS A. If $A \models \varphi$ then $S \models \varphi$.*

The theorem states that when model checking of a particular property on a RATS returns true, we are sure that the property is satisfied in the original dynamic system. However, when the result is negative, the counterexample returned does not necessarily reflect any trajectory in the original system.

The system in Figure 5 satisfies a formula $\mathbf{FG}(B \leq 3)$ expressing the temporal property stating that despite the choice of the initial state the system eventually stabilises at states where concentration of B is kept below 3. Now let us consider a formula $\mathbf{F}(B > 3)$ expressing the property that despite the initial settings, the concentration of B will eventually exceed the concentration level 3. In this case the model checking returns one of the counterexamples as emphasised in Figure 6 stating that if initially $A < 5$ and $B < 3$ then B is not increased while staying indefinitely long in the shaded state.

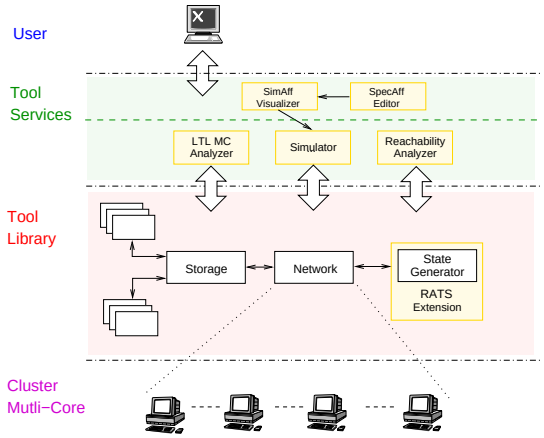


Figure 7: BioDiVinE Toolset Architecture

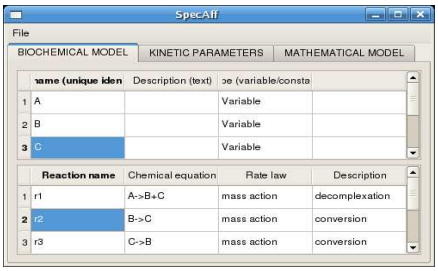


Figure 8: A biochemical model specified in BioDiVinE GUI

3 BioDiVinE Tool

BioDiVinE employs aggregate power of network-interconnected workstations (nodes) to analyse large-scale state transition systems whose exploration is beyond capabilities of sequential tools. System properties can be specified either directly in Linear Temporal Logic (LTL) or alternatively as processes describing undesired behaviour of systems under consideration (negative claim automata). From the algorithmic point of view, the tool implements a variety of novel parallel algorithms [9, 1] for cycle detection (LTL model checking). By these algorithms, the entire state space is uniformly split into partitions and every partition is distributed to a particular computing node. Each node is responsible for generating the respective state-space partition on-the-fly while storing visited states into the local memory.

The state space generator constructs the rectangular abstraction transition system for a given multi-affine system. The scheme of the tool architecture is provided in Figure 7. Library-level components are responsible for constructing, managing and distributing the state space. They form the core of the tool. The tool provides two graphical user interface components *SpecAff* — allowing editing of biological models in terms of chemical reactions, and *SimAff* — allowing visualisation of the simulation results.

The input (biochemical) model is given by the following data:

- list of chemical species,
- list of partitioning thresholds given for each species,
- list of chemical reactions.

```
VARS:A,B,C

EQ:dA = (-0.1)*A
EQ:dB = 0.1*A + (-1)*B + 1*C
EQ:dC = 0.1*A + 1*B + (-1)*C

TRES:A: 0, 4, 6, 10
TRES:B: 0, 2, 10
TRES:C: 0, 2, 4, 10

INIT:   4:6, 0:2, 2:4
```

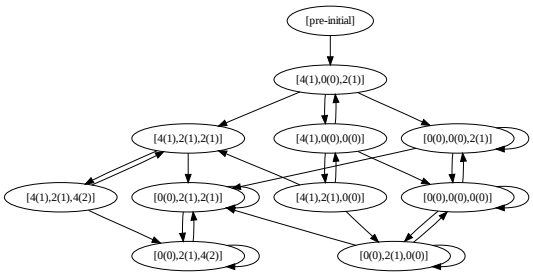


Figure 9: A multi-affine ODE model and its state space generated by BioDiVinE

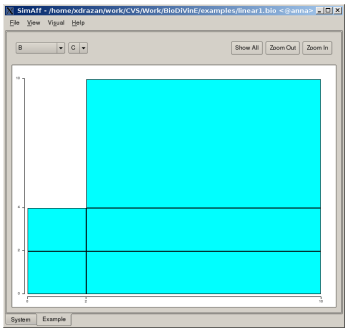


Figure 10: Visualisation of the state space in the BioDiVinE GUI, states are projected onto the BC concentration plane

The biochemical model together with the tested property and initial conditions are then automatically translated into a multi-affine system of ODEs forming the mathematical model that can be analysed by BioDiVinE algorithms. The mathematical model consists of the following data:

- list of variables,
- list of (multi-affine) ODEs,
- list of partitioning thresholds given for each species,
- list of initial rectangular subspaces (the union of these subspaces forms the initial condition),
- Büchi automaton representing an LTL property (this data is not needed for simulation).

An example of a simple three-species model representing three biochemical reactions $A \rightarrow B + C$, $B \rightleftharpoons C$ is showed in Figure 8. The first reaction is performed at rate 0.1 s^{-1}. The second two reversible reactions are at rate 1 s^{-1}. The respective mathematical model is showed in Figure 9 on the left in the textual `.bio` format. For each variable there is specified the equation as well as the list of real values representing individual threshold positions. The initial condition is defined in this particular case by a single rectangular subspace: $A \in \langle 4,6\rangle, B \in \langle 0,2\rangle$, and $C \in \langle 2,4\rangle$. The state space generated for this setting is depicted in Figure 9 on the right. Figure 10 demonstrates visualisation features of the BioDiVinE GUI.

For model checking analysis, BioDiVinE relies on the parallel LTL model checking algorithms of the underlying DiVinE library [4]. A given LTL formula is translated into a Büchi automaton which

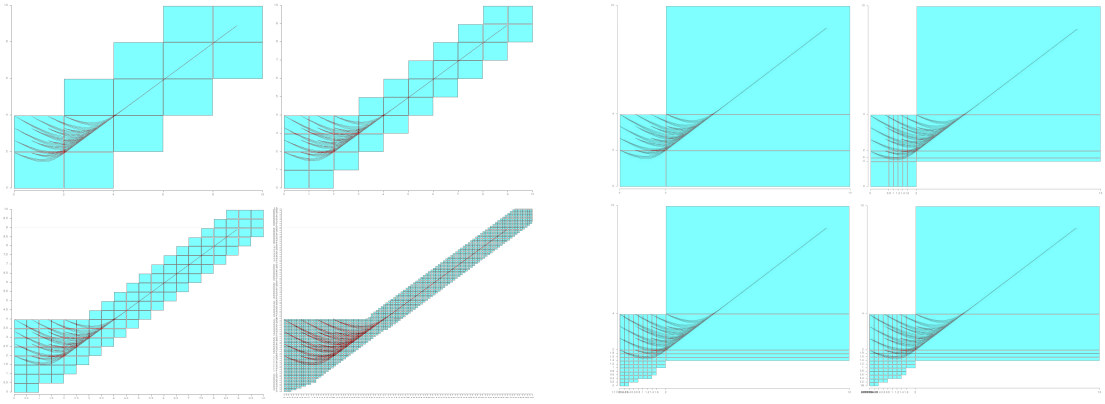


Figure 11: A visualisation of the state-spaces in BC projection with uniform (left) and automatic (right) gradual threshold refinement, manually augmented by numeric simulation trace from COPASI [11]

```
VARS:A,B,C

EQ:dA = (-0.1)*A
EQ:dB = 0.1*A + (-1)*B + 1*C
EQ:dC = 0.1*A + (1)*B + (-1)*C

TRES:A: 0, 4, 6, 10
TRES:B: 0, 2, 10
TRES:C: 0, 2, 4, 10

INIT:   4:6, 0:2, 2:4
```

```
process LTL_property1 {
state q1, q2;
init q1;
accept q2;
trans
q1 -> q2 { guard B>4 && B<4.5
                && C>3 && C<3.5; },
q1 -> q1 {},
q2 -> q2 {};
}

system sync property LTL_property1;
```

```
process LTL_property2 {
state q1, q2;
init q1;
accept q2;
trans
q1 -> q2 { guard B>4 && B<4.5
                && C>5.5 && C<6; },
q1 -> q1 {},
q2 -> q2 {};
}

system sync property LTL_property2;
```

Figure 12: A multi-affine model extended with a never claim automaton for property 1 and property 2

represents its negation. That way the automaton represents the never claim property. The automaton is automatically generated for an LTL formula and merged with the mathematical model by the `divine.combine` utility. An example of a model extended with a never claim property is showed in Figure 12. In particular, the first automaton `LTL_property1` specified in terms of the DiVinE language represents a never claim for the safety LTL formula $\mathbf{G}\neg(B \geq 4 \wedge B \leq 4.5 \wedge C \geq 3 \wedge C \leq 3.5)$ expressing that concentrations of species B and C will never enter the specified rectangular region. The unreachability of a slightly different region is defined by the automaton for property 2. The results of the model-checking procedure are showed in Figure 13. Property 1 has been proved false by finding a run of the system visiting the specified region (states of run given as list), while property 2 has been proved true by extensively searching all the system runs and not finding any one that would cross the region specified in property 2.

The choice of threshold values for each variable affects greatly the shape and size of the generated state-space. The refinement of a given partitioning — the addition of more thresholds to a set of former thresholds — may result in the unreachability of a part of a region reachable before.

Since manual refinement of thresholds by adding numeric values can be tedious or unintuitive, two more advanced methods are available in BioDiVinE. The first method divides a given interval uniformly into subintervals of a given size, while the second method tries a more sophisticated automatic technique of dividing regions according to signs of the concentration derivatives inside these regions [17]. The

```
=======================
--- Accepting cycle ---
=======================
[pre-initial]
[4(1),0(0),2(1)-PP:0]
[4(1),2(1),2(1)-PP:0]
[4(1),2(1),3(2)-PP:0]
[4(1),4(2),3(2)-PP:1]
[0(0),4(2),3(2)-PP:1]
======= Cycle =======
[0(0),2(1),3(2)-PP:1]
[0(0),2(1),2(1)-PP:1]
```

```
============================
  --- No accepting cycle ---
============================
states:            33
transitions:       81
iterations:        1
size of a state:   16
size of appendix:  12
cross transitions: 0
all memory         56.5 MB
time:              0.115177 s
--------------------
0: local states:      33
0: local memory:      56.5
```

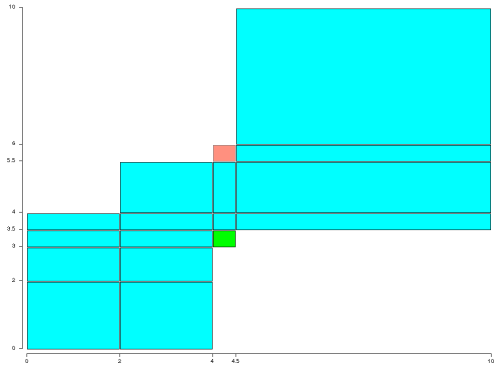


Figure 13: Results of model-checking for property 1 (left), property 2 (middle), visualisation of reachable (green) and unreachable (red) regions

k	States	Trans	Time on particular number of cluster processor cores (s)																		
			1	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72
5	$3 \cdot 10^4$	$8.5 \cdot 10^4$	15.4	4.9	2.7	1.8	1.4	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
	relative	speedup	1	3.14	5.7	8.56	11	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
10	$9 \cdot 10^5$	$3.2 \cdot 10^6$	$\top$	$\top$	$\top$	161	119	107	85	72	63	55	53	46	44	40	38	38	$\perp$	$\perp$	$\perp$
	relative	speedup	$\top$	$\top$	$\top$	1	1.35	1.5	1.89	2.24	2.56	2.93	3.04	3.5	3.66	4.03	4.24	4.24	$\perp$	$\perp$	$\perp$
15	$1.6 \cdot 10^6$	$6.5 \cdot 10^6$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	222	204	177	156	146	129	122	117	110	98	93
	relative	speedup	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	1	1.09	1.25	1.42	1.52	1.72	1.82	1.9	2.02	2.27	2.39
20	$3.2 \cdot 10^6$	$1.4 \cdot 10^7$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	202	180	173
	relative	speedup	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	1	1.12	1.17

Figure 14: Scaling of model checking algorithms on a homogeneous cluster

state-spaces resulting from the gradual application of both threshold refinement approaches are depicted in Figure 11. It is important to mention that the overall size of the state-space depends exponentially on the number of thresholds for all species. However, in some cases the actual reachable subspace of the whole state-space may be only polynomial in the number of thresholds.

For any multi-affine model extended with a never claim automaton as showed in Figure 12, the parallel model checking algorithms can be directly called.

We have performed several experiments [2] in order to show scaling of the algorithms when distributed on several cluster nodes. Figures 14 and 15 show scaling of model checking conducted on a simple model of a reaction network representing a catalytic reaction scaled for different numbers of intermediate products.

4 Case Study: Ammonium Transport in E. Coli

In this section we present a case study conducted using the current version of BioDiVinE. Since the rectangular abstraction method of multi-affine systems implemented in BioDiVinE is a relatively new result of applied control theory, its application is still in the stage of experimentation. In fact, we are still unaware of any case studies that apply this method to real biological models. In this case study we focus on demonstrating the usability of rectangular abstraction to analysis of biological models. To this end, we consider a simple biological model that specifies ammonium transport from external environment into the cells of Escherichia Coli. This simplified model is based on a published model of the E. Coli ammonium assimilation system [19] which was developed under the EC-MOAN project (`http://www.ec-moan.org`). The metabolic reactions and regulatory reactions in the original model were removed.

$$S + E \rightleftharpoons ES_1 \rightleftharpoons ES_2 \rightleftharpoons \cdots \rightleftharpoons ES_k \rightarrow P + E$$

$$\begin{aligned}
\dot{S} &= ES_1 - 0.01 \cdot E \cdot S \\
\dot{E} &= ES_1 - 0.01 \cdot S \cdot E + ES_1 \\
\dot{ES_1} &= 0.01 \cdot E \cdot S - ES_1 - ES_1 \\
\dot{ES_2} &= \cdot ES_1 - 2 \cdot ES_2 + ES_3 \\
&\vdots \\
\dot{ES_{k-1}} &= ES_{k-2} - 2 \cdot ES_{k-1} + ES_k \\
\dot{ES_k} &= ES_{k-1} - 2 \cdot ES_k
\end{aligned}$$

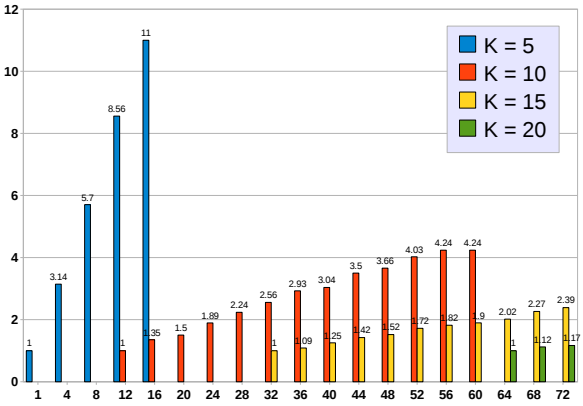


Figure 15: Scaling of model checking algorithms on a homogeneous cluster, X-axis shows number of cluster nodes, Y-axis gives speedup relative to system size and number of nodes

4.1 Model Description

E. coli can express membrane bound transport proteins for the transportion of small molecules from the environment into the cytoplasm at certain conditions. At normal ammonium concentration, the free diffusion of ammonia can provide enough flux for the growth requirement of nitrogen. When ammonium concentration is very low, E. coli cells express *AmtB* (an *ammonia transporter*) to complement the deficient diffusion process. Three molecules of *AmtB* (trimer) form a channel for the transportation of ammonium/ammonia. Protein structure analysis revealed that *AmtB* binds $NH4^+$ at the entrance gate of the channel, deprotonates it and conducts NH_3 into the cytoplasm as illustrated in Figure 16 (left) [16]. At the periplasmic side of the channel there is a wider vestibule site capable of recruiting NH_4^+ cations. The recruited cations are passed through the hydrophobic channel where the pKa of NH_4^+ was shifted from 9.25 to below 6, thereby shifting the equilibrium toward the production of NH_3. NH_3 is finally released at the cytoplasmic gate and converted to NH_4^+ because intracellular pH (7.5) is far below the pKa of NH_4^+.

In addition to the above mentioned *AmtB* mediated transport, the bidirectional free diffusion of the uncharged ammonia through the membrane is also included in the simplified model. The intracellular NH_4^+ is then metabolised by Glutamine Synthetase (GS). The whole model is depicted in Figure 16 (right). The external ammonium is represented in the uncharged and charged forms denoted NH_3ex and NH_4^+ex. Analogously, the internal ammonium forms are denoted NH_3in and NH_4^+in. The biochemical model that combines *AmtB* transport with NH_3 diffusion is given in Table 1. The kinetic parameters are based on the values in the original model.

By employing the law of mass action kinetics the reaction network is transformed into the set of multi-affine ODEs as listed in Table 2. Since we are especially interested in how the concentrations of internal ammonium change with respect to the external ammonium concentrations, we employ the following simplifications:

- We do not consider the dynamics of the external ammonium forms, thus we take NH_3ex and NH_4^+ex as constants (the input parameters for the analysis).

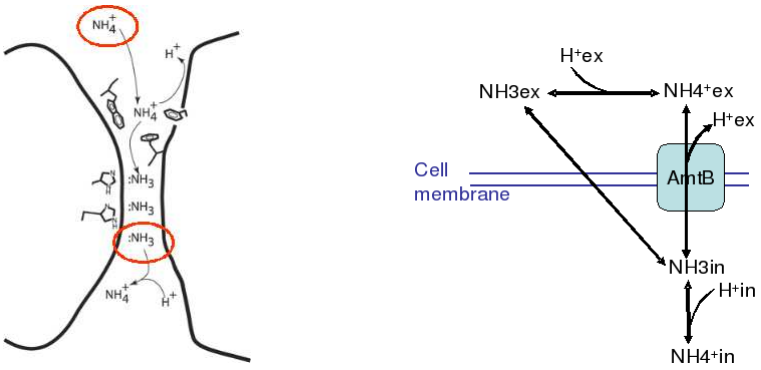


Figure 16: E. Coli ammonium transport mechanism and the respective pathway

$$
\begin{array}{rl}
AmtB + NH_4ex \overset{k_1}{\underset{k_2}{\longleftrightarrow}} AmtB:NH_4 & k_1 = 5\cdot 10^8, k_2 = 5\cdot 10^3 \\
AmtB:NH_4 \overset{k_3}{\longrightarrow} AmtB:NH_3 + H_{ex} & k_3 = 50 \\
AmtB:NH_3 \overset{k_4}{\longrightarrow} AmtB + NH_3in & k_4 = 50 \\
NH_4in \overset{k_5}{\longrightarrow} & k_5 = 80 \\
NH_3in + H_{in} \overset{k_6}{\underset{k_7}{\longleftrightarrow}} NH_4in & k_6 = 1\cdot 10^{15}, k_7 = 5.62\cdot 10^5 \\
NH_3ex \overset{k_8}{\underset{k_9}{\longleftrightarrow}} NH_3in & k_8 = k_9 = 1.4\cdot 10^4
\end{array}
$$

Table 1: Biochemical model of ammonium transport

- We assume constant intracellular pH (7.5) and extracellular pH (7.0), thus H_{ex} and H_{in} are calculated to be $3\cdot 10^{-8}$ and 10^{-7}. Based on the extracellular pH and the total ammonium/ammonia concentration, concentrations of NH_3ex and NH_4^+ex can be calculated.

Without the loss of correctness, we simplify the notation of the cation NH_4^+ as NH_4.

4.2 Model Analysis

From the essence of biophysical laws, it is clear that the maximal reachable concentration level accumulated in the internal ammonium forms directly depends on the ammonium sources available in the environment. However, it is not directly clear what particular maximal level of internal ammonium is achievable at given amount of external ammonium (distributed into the two forms). In the analysis we have focused on just this phenomenon. More precisely, the problem to solve has been to analyse how

$$
\begin{aligned}
\tfrac{dAmtB}{dt} &= -k_1\cdot AmtB\cdot NH_4ex + k_2\cdot AmtB:NH_4 + k_4\cdot AmtB:NH_3 \\
\tfrac{dAmtB:NH_3}{dt} &= k_3\cdot AmtB:NH_4 - k_4\cdot AmtB:NH_3 \\
\tfrac{dAmtB:NH_4}{dt} &= k_1\cdot AmtB\cdot NH_4ex - k_2\cdot AmtB:NH_4 - k_3\cdot AmtB:NH_4 \\
\tfrac{dNH_3in}{dt} &= k_4\cdot AmtB:NH_3 - k_7\cdot NH_3in + k_6\cdot NH_4in \\
\tfrac{dNH_4in}{dt} &= k_5\cdot NH_4in + k_7\cdot NH_3in\cdot H_{in} - k_6\cdot NH_4in
\end{aligned}
$$

Table 2: Mathematical model of ammonium transport

Partition	$AmtB$	$AmtB:NH_3$	$AmtB:NH_4$	NH_3in	NH_4in
(1)	7	3	3	6	4
(2)	7	9	3	8	26

Table 3: Numbers of thresholds in partitions (1) and (2)

the setting of the model parameters NH_3ex and NH_4^+ex affects the maximal concentration level of NH_3in and NH_4^+in reachable from given initial conditions.

During discussions with biologists we have found out that there is currently not available any in vitro measurement of the $AmtB$-concentration (and also the concentrations of dimers $AmtB:NH_3$ and $AmtB:NH_4$). Hence there has appeared the need to analyse the model with uncertain initial conditions. Such a setting fits the current features of BioDiVinE, especially the rectangular abstraction that naturally abstracts away the exact concentration values up to the intervals between certain concentration levels.

4.2.1 Maximal reachable levels of internal ammonium forms

At first, we have considered the (abstracted) initial condition to be set to the following intervals between concentration values:

$$AmtB \in \langle 0, 1\cdot 10^{-5}\rangle, \quad AmtB:NH_3 \in \langle 0, 1\cdot 10^{-5}\rangle, \quad AmtB:NH_4 \in \langle 0, 1\cdot 10^{-5}\rangle,$$
$$NH_3in \in \langle 1\cdot 1^{-6}, 1.1\cdot 10^{-6}\rangle, \quad NH_4in \in \langle 2\cdot 10^{-6}, 2.1\cdot 10^{-6}\rangle$$

Note that the upper bounds as well as the initial intervals of internal ammonium forms have been set with respect to the available data obtained from the literature.

We have considered two rectangular partitions. The partition (1) has been set basically according to the initial conditions. The partition (2) has been obtained by running one iteration of the automatic threshold refinement procedure to partition (1). Numbers of thresholds per each variable are given in Table 3.

We have conducted several model checking experiments in order to determine the maximal reachable concentration levels of NH_3in and NH_4^+in. In particular, we have searched for the lowest α satisfying the property $\mathbf{G}(NH_3in < \alpha)$ and the lowest β satisfying $\mathbf{G}(NH_4in < \beta)$. The property $\mathbf{G}p$ requires that all paths available in the rectangular abstraction from the states specified by the initial condition must satisfy the given proposition p at every state. Note that if the model checking method finds the property $\mathbf{G}p$ false in the model, it also returns a counterexample for that. The counterexample satisfies the negation of the checked formula, which is in this case $\mathbf{F}\neg p$. Interpreting this observation intuitively for the given formulae, we use model checking to find a path on which the species NH_3in (resp. NH_4in) exceeds the level α (resp. β).

We did not want to get precise values of α, β, but we rather wanted to get their good approximation. At the starting point, we substituted for α (resp. β) the upper initial bounds of the respective variables. Then we found the requested values by iteratively increasing and decreasing α (resp. β). The obtained results are summarised in Table 4.

The results have shown that NH_3in does not exceed its initial level no matter how the external ammonium is distributed between NH_3ex and NH_4^+ex. The upper bound concentration considered for both NH_3ex and $NH4^+ex$ has been set to $1\cdot 10^{-5}$ which goes with the typical level of concentration of the gas in the environment.

In the case of NH_4in we have found that the upper bound to maximal reachable level is in the interval $\beta \in \langle 5.3\cdot 10^{-4}, 5.4\cdot 10^{-4}\rangle$. Since the counterexample achieved can be a spurious one due to the

Partition (1)				Partition (2)		
α	$\mathbf{G}(NH_3in < \alpha)$	# states	Time (# nodes)	$\mathbf{G}(NH_3in < \alpha)$	# states	Time (# nodes)
$1.1 \cdot 10^{-6}$	true	1081	0.36 s (1)	true	$1.5 \cdot 10^5$	1.9 s (18)
β	$\mathbf{G}(NH_4in < \beta)$	# states	Time (# nodes)	$\mathbf{G}(NH_4in < \beta)$	# states	Time (# nodes)
$1 \cdot 10^{-3}$	true	2161	0.45 s (1)	true	$1.6 \cdot 10^5$	2 s (18)
$5 \cdot 10^{-4}$	false	4753	1.9 s (1)	false	$2.7 \cdot 10^5$	3 s (18)
$6 \cdot 10^{-4}$	true	2161	0.43 s (1)	true	$1.5 \cdot 10^5$	1.8 s (18)
$5.4 \cdot 10^{-4}$	true	1441	0.27 s (1)	true	$2.1 \cdot 10^5$	4.2 s (18)
$5.3 \cdot 10^{-4}$	false	3421	1.2 s (1)	false	$2.7 \cdot 10^5$	2.2 s (18)

Table 4: Experiments on detecting maximal reachable levels of internal ammonium forms

Partition (1)				Partition (2)		
NH_3ex	φ	# states	Time (# nodes)	φ	# states	Time (# nodes)
$19.5 \cdot 10^{-4}$	true	901	0.22 s (1)	true	$1.4 \cdot 10^5$	1.9 s (36)
$19.6 \cdot 10^{-4}$	false	1261	0.6 s (1)	false	$3.4 \cdot 10^5$	5.9 s (36)

Table 5: Experiments on detecting NH_3ex levels affecting maximal reachable NH_3in

overapproximating abstraction, the exact maximal reachable value may be lower. To this end we have conducted several numerical simulations which give us the argument that our estimation of β is plausible.

4.2.2 Dependence of stable internal ammonium on changes in external conditions

In the second experiment, we have focused on determining how much external ammonium have to be increased in particular form in order to stimulate $NH3in$ to exceed the considered initial level. The setting of partitions and initial conditions has been considered the same as in the previous experiments.

First, we have varied the constant amount of NH_3ex to find at which level of NH_3ex the maximal reachable level of NH_3in is affected. More precisely, we have observed for which setting of NH_3ex the property $\varphi \equiv \mathbf{G}(NH_3in < 1.1 \cdot 10^{-6})$ is true and for which it is not. The relevant experiments are summarised in Table 5. We have found out that if NH_3ex is set to $19.6 \cdot 10^{-4}$ or higher level then NH_3in increases above the upper initial bound $1.1 \cdot 10^{-6}$. The counterexamples returned again agree with numerical simulations.

Finally, we have varied the amount of NH_4^+ex in order to find at which level of NH_4^+ex the maximal reachable level of NH_3in is affected. The results presented in Table 6 give us the conclusion that despite the level of NH_4^+ex (checked up to 10^{12}), the maximal level reached by NH_3in remains the same. In particular, NH_3in does not exceed the initial bounds.

Partition (1)			
NH_4ex	φ	# states	Time (# nodes)
1	true	901	0.25 s (1)
$1 \cdot 10^{12}$	true	901	0.25 s (1)

Table 6: Experiments on detecting NH_4^+ex levels affecting maximal reachable NH_3in

4.2.3 Performance

All the experiments have been performed on a homogeneous cluster allowing computation on up-to 22 nodes each equipped with 16GB of RAM and a quad-core processor Intel Xeon 2GHz. The model we have dealt with contained 5 dynamic variables and 2 constants. With partition (1) the generated state space had maximally 4753 states and with partition (2) maximally $3.4 \cdot 10^5$. When trying to run one more step of automatic partition refinement, the number of thresholds exceeded the memory reserved for storing of the mathematical model. Note that the model has to be stored in the memory of each node. Only the state space is distributed over the cluster.

5 Conclusion

In this paper, we have presented the current version of BioDiVinE tool which implements rectangular abstraction of continuous models of biochemical reaction systems. The tool provides the framework for specification and analysis of biochemical systems. The supported analysis technique is based on the model checking method. Linear temporal logic is used for encoding of the properties to be observed on abstracted systems.

The tool provides parallel model checking algorithms that allow fast response times of the analysis. We have provided a case study on which the key features of the tool are demonstrated. The case study has showed that the tool can be used for quickly getting the approximation of maximal reachable concentration levels of individual species in the model. In general, we have analysed the model with respect to the set of safety properties which are technically tackled by construction of the state space reachable from the given initial states. We have found out that the main advantage of the rectangular abstraction is the possibility to analyse the system with uncertain initial conditions.

The current drawback of the abstraction method is strong overapproximation of non-transient states in multi-affine systems. In consequence, analysis of liveness kind of properties (e.g., oscillations, instability) is infeasible because of large amount of spurious counterexamples that come from false identification of non-transient states. However, this is not the case for affine systems on which liveness properties can be checked without these problems. Since the analysed systems are typically non-affine, we can still employ the liveness checking on their linearised approximations. However, by the linearisation process the precision of the analysis is significantly reduced. Improving the tool in these aspects requires further research.

In general, we leave for future work the development of methods for identification of spurious paths. We think that one of the promising directions in using the discrete abstractions for analysis of biological models is employing the model-checking-based analysis for extensive exploration of properties. In particular, instead of returning only one path, the model checker should provide a set of paths. In this directions we aim to continue the research.

References

[1] J. Barnat, L. Brim & J. Chaloupka (2005): *From Distributed Memory Cycle Detection to Parallel LTL Model Checking*. *Electronic Notes in Theoretical Computer Science* 133(1), pp. 21–39.

[2] J. Barnat, L. Brim, I. Černá, S. Dražan, J. Fabriková & D. Šafránek (2009): *Computational Analysis of Large-Scale Multi-Affine ODE Models* Accepted to High Performance Computational Systems Biology (HIBI) 2009.

[3] J. Barnat, L. Brim, I. Černá, S. Dražan, J. Fabriková & D. Šafránek (2009): *On Algorithmic Analysis of Transcriptional Regulation by LTL Model Checking*. Theoretical Computer Science 410, pp. 3128–3148.

[4] J. Barnat, L. Brim, I. Černá, P. Moravec, P. Ročkai & P. Šimeček (2006): *DiVinE – A Tool for Distributed Verification (Tool Paper)*. In: Computer Aided Verification, LNCS 4144/2006. Springer, pp. 278–281.

[5] G. Batt, C. Belta & R. Weiss (2007): *Model checking liveness properties of genetic regulatory networks*. In: Conference on Tools and Algorithms for the Construction and Analysis of Systems, LNCS 4424. Springer, pp. 323–338.

[6] G. Batt, C. Belta & R. Weiss (2008): *Model checking genetic regulatory networks with parameter uncertainty*. IEEE Transactions on Circuits and Systems and IEEE Transactions on Automatic Control, Special Issue on Systems Biology , pp. 215–229.

[7] C. Belta & L.C.G.J.M. Habets (2006): *Controlling a class of nonlinear systems on rectangles*. IEEE Transactions on Automatic Control 51(11), pp. 1749–1759.

[8] L. Calzone, F. Fages & S. Soliman (2006): *BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge*. Bioinformatics 22(14), pp. 1805–1807.

[9] I. Černá & R. Pelánek (2003): *Distributed Explicit Fair Cycle Detection (Set Based Approach)*. In: Model Checking Software. 10th International SPIN Workshop, LNCS 2648. Springer, pp. 49 – 73.

[10] E.M. Clarke, O. Grumberg & D.A. Peled (2000): *Model Checking*. MIT Press.

[11] S. Hoopes et.al. (2006): *COPASI – a COmplex PAthway SImulator*. Bioinformatics 22, pp. 3067–74.

[12] M. Feinberg (1987): *Chemical reaction network structure and the stability of complex isothermal reactors I. The deficiency zero and the deficiency one theorems*. Chemical Engineering Science 42, pp. 2229–2268.

[13] S.K. Jha, E.M. Clarke, C.J. Langmead, A. Legay, A. Platzer & A.P. Zuliani (2009): *A Bayesian Approach to Model Checking Biological Systems*. In: Proc. of Computational Methods in Systems Biology (CMSB 2009), 5688. Springer, pp. 218–234.

[14] H. de Jong, J. Geiselmann, C. Hernandez & M. Page (2003): *Genetic Network Analyzer: qualitative simulation of genetic regulatory networks*. Bioinformatics 19(3), pp. 336–344. Available at `http://www-helix.inrialpes.fr/gna`.

[15] H. de Jong, J.L. Gouzé, C. Hernandez, M. Page, T. Sari & J. Geiselmann (2004): *Qualitative simulation of genetic regulatory networks using piece-wise linear models*. Bull. Math. Biology 66, pp. 301–340.

[16] S. Khademi, J. O'Connell III, J. Remis, Y. Robles-Colmenares, L.J. Miercke & R.M. Stroud (2004): *Mechanism of ammonia transport by Amt/MEP/Rh: Structure of AmtB at 1.35*. Science 305(5690), pp. 1587–1594.

[17] M. Kloetzer & C. Belta (2008): *Reachability analysis of multi-affine systems*. Transactions of the Institute of Measurement and Control In press.

[18] X. Liu, J. Jiang, O. Ajayi, X. Gu, D. Gilbert & R. Sinnott (2008): *BioNessie*. Global Healthgrid: e-Science Meets Biomedical Informatics - Proceedings of HealthGrid 2008 138, pp. 147–157.

[19] H. Ma, F. Boogerd & I. Goryanin (2009): *Modelling nitrogen assimilation of Escherichia coli at low ammonium concentration*. Journal of Biotechnology Doi:10.1016/j.jbiotec.2009.09.003.

A study on the combined interplay between stochastic fluctuations and the number of flagella in bacterial chemotaxis

Daniela Besozzi[a] Paolo Cazzaniga[b] Matteo Dugo[a]
Dario Pescini[b] Giancarlo Mauri[b]

[a]Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39, 20135 Milano, Italy
besozzi@dico.unimi.it, matteo.dugo@studenti.unimi.it

[b]Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Viale Sarca 336, 20126 Milano, Italy
cazzaniga/pescini/mauri@disco.unimib.it

The chemotactic pathway allows bacteria to respond and adapt to environmental changes, by tuning the tumbling and running motions that are due to clockwise and counterclockwise rotations of their flagella. The pathway is tightly regulated by feedback mechanisms governed by the phosphorylation and methylation of several proteins. In this paper, we present a detailed mechanistic model for chemotaxis, that considers all of its transmembrane and cytoplasmic components, and their mutual interactions. Stochastic simulations of the dynamics of a pivotal protein, CheYp, are performed by means of tau leaping algorithm. This approach is then used to investigate the interplay between the stochastic fluctuations of CheYp amount and the number of cellular flagella. Our results suggest that the combination of these factors might represent a relevant component for chemotaxis. Moreover, we study the pathway under various conditions, such as different methylation levels and ligand amounts, in order to test its adaptation response. Some issues for future work are finally discussed.

1 Introduction

Recent experimental investigations at the single-cell level [9] have evidenced the presence of biological noise, due to the inherently stochastic interactions between those molecular species that occur in low amounts inside the cell. Standard modeling approaches based on ordinary differential equations cannot effectively capture the effects of biological random processes, which are able to lead the cell to different states (e.g., bistability). In the last years, indeed, many algorithms that perform stochastic simulations of biochemical reaction systems have proved their intrinsic suitability for reproducing the dynamics of many cellular processes (see, e.g., [24] and references therein). For instance, the stochastic simulation algorithm (SSA) [10] is able to provide an exact realization of the system dynamics and to account for random fluctuations in the temporal evolution of molecules; however, it can be very slow for those systems where the number of reactions – or even the amount of a few molecular species – is large, as it is frequently the case for complex cellular processes involving many species and many interactions among them. In order to overcome this drawback, a faster and reliable stochastic simulation algorithm, called tau leaping [4], has been recently proposed. Tau leaping represents an efficient tool for the modeling of

R.J.Back, I.Petre, E. de Vink (Eds.): Computational
Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 47–62, doi:10.4204/EPTCS.6.4

stochastic processes in individual cells (see, e.g., [6]), as it can easily handle very detailed and mechanistic descriptions of all possible molecular interactions or molecule modifications, which can bring about a huge increase in the number of reactions and molecular species (e.g., phosphorylation or methylation states of proteins). In this work we exploit the efficiency of tau leaping to simulate the dynamics of bacterial chemotaxis, in order to generate stochastic time series of a pivotal chemotactic protein, which will be then analyzed with respect to the number of flagella in bacterial cells.

Chemotaxis is an efficient signal transduction pathway which allows bacterial cells to move directionally, in response to specific attractants or repellents occurring in their surroundings. The pathway consists of several transmembrane and cytoplasmic proteins acting as signal sensors and response regulators [11], which rule the reversal of the flagellar motor (governed by the phosphorylation and dephosphorylation of a key protein, CheY). This process induces a switch between running and tumbling movements, with a frequency that allows a temporal sampling (through random walks) of homogeneous environments. Anyway, in the presence of a gradient of attractants or repellents, the bacteria are able to respond quickly by reducing the frequency of flagellar reversal between clockwise and counterclockwise rotations, which cause a longer running motion in a biased direction. The frequency of switching is then reset to the random walk level if the concentration of the external ligands remains constant in time. At the molecular scale, this adaptation property is implemented by the coordinated action of methyltransferase and methylesterase proteins acting on the transmembrane receptors.

The genetic regulation and biochemical functions of the proteins involved in chemotaxis are well known, and several models have already been proposed to study their complex interplay as well as the robustness of this system [1, 23, 17, 13, 15, 19]. In the model we present hereby, we consider very detailed protein-protein interactions for the chemotactic pathway in *E. coli*, in response to attractant molecules, which sum up to 62 biochemical reactions and 32 molecular species. The temporal evolution of the phosphorylated form of CheY (CheYp) is investigated under different conditions, such as the deletion of other proteins involved in the pathway, the addition of distinct amounts of external ligand, and the effect of different methylation states.

The results obtained through stochastic simulations of this model are then used to propose an analysis on the interplay between the stochastic fluctuations of CheYp and the number of cellular flagella, which occur in a few units in the individual bacterium (around half a dozen in *E. coli*). The aim of this analysis is to devise the mean time periods during which the cell either performs a running or a tumbling motion, considering both the coordination of flagella and the randomness that is intrinsic in the chemotactic pathway. Indeed, experimental observations show that the running motion requires all flagella to be simultaneously synchronized in the counterclockwise rotation, which occurs when CheYp is not interacting with the proteins regulating the flagellar motor; on the contrary, when at least one flagellum is not coordinated with the others, then the bacterium performs a tumbling movement. To distinguish between these two states, we will assume that the cell is sensitive to a threshold level of CheYp, that is evaluated as the mean value of CheYp at steady state. Because of stochastic fluctuations, the amount of CheYp will randomly switch from below to above this value, thus reversing the rotation from counterclockwise to clockwise rotations of each flagellum. Therefore, the original contribution of our work consists in linking the synchronization of all flagella to the stochastic fluctuations of CheYp, as the core component that stands at the basis of chemotactic motions. To this aim, we define a procedure to identify the synchronization of rotations of all flagella, and we use it to compare the mean time intervals of running and tumbling motions – as well as of the adaptation times after ligand addition – according to a varying number of flagella.

The paper is structured as follows. In Section 2 we give a description of the bacterial chemotactic pathway, and present the mechanistic model of this system. In Section 3, after a brief explanation of

the functioning of tau leaping, we show the results of simulations for the temporal evolution of the pivotal protein in chemotaxis (CheYp), that have been obtained by using the model previously defined. Then, in Section 4 we exploit the outcome of stochastic simulations to study the relations between the fluctuations of CheYp and the number of flagella occurring in a bacterial cell. We conclude the paper with some topics for future extensions of our work.

2 The modeling of bacterial chemotaxis

In this section we present the chemotaxis signaling pathway and define the mechanistic model that describes the molecular interactions therein involved.

2.1 Bacterial chemotaxis

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

The chemotactic pathway, depicted in Figure 1, has been well characterized from a molecular point of view [3, 11, 25]. External signals are detected by transmembrane methyl-accepting proteins (MCPs), which are linked to cytoplasmic histidine protein kinases (CheA) by means of scaffold proteins (CheW). These three proteins constitute the sensor module (i.e. the receptor complexes) of the whole pathway; each protein occurs as a dimer in every receptor complex. The role of CheA is to transduce the presence of an external ligand toward the inside of the cell, by phosphorylating two cytoplasmic proteins, called CheY and CheB. The transfer of the phosphoryl group to these proteins is more probable – that is, the activity of CheA is stronger – in absence of external ligands. CheY and CheB compete for the binding to CheA, but the phosphotransfer to CheY is faster than to CheB [25]; this fact assures that the proper chemotactic response can be generated before the process of adaptation occurs, as explained hereafter. CheY is the response regulator protein which, after being phosphorylated, interacts with the proteins FliM of the flagellar motors, inducing the CW rotation of the flagellum and the tumbling movements (FliM is a key component of the processes that stand *downstream* of the chemotaxis signaling, and therefore will not be explicitly included in our model; anyway, some considerations about its role within the model are discussed in Section 5). In presence of external ligands, the activity of CheA is reduced: the

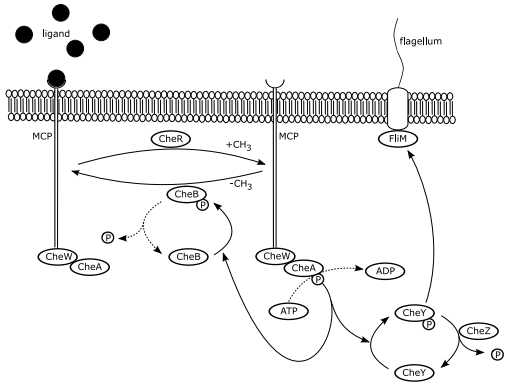


Figure 1: Signal transduction pathway in bacterial chemotaxis: solid arrows indicate enzyme-catalyzed reactions, dashed arrows indicate autocatalysis; CH_3 denotes the methyl group, P the phosphoryl group (the dimensions of components are not scaled).

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

2.2 A mechanistic model

For the modeling of the chemotaxis pathway, we have considered detailed protein-protein interactions which sum up to a total of 62 reactions and 32 molecular species [7]. The initial amounts – given as number of molecules, as reported in [21] – of the 7 elementary chemotactic proteins are the following: 4000 dimers of MCPs; 4000 dimers of CheW; 4000 dimers of CheA; 17000 monomers of CheY; 12000 monomers of CheZ; 200 monomers of CheR; 1700 monomers of CheB (plus a constant amount of $1.2 \cdot 10^6$ ATP molecules that are needed for phosphorylation reactions). The initial amounts of all other molecular species appearing in the model are equal to zero, as they are produced by mimicking the formation and dissociation of protein complexes, and by describing the phosphorylation/dephosphorylation of cytoplasmic proteins and the methylation/demethylation of MCPs, in both the conditions of presence and absence of external ligands.

Each reaction in the model is given in the form "reagents $\rightarrow$ products", where the notation X + Y is used to represent a molecular interaction between species X and Y, while X::Y denotes that X and Y are chemically bound in the formation of a complex (see Table 1). Note that only monomolecular or bimolecular reactions are here considered; the formation of complexes consisting of more than two species is performed stepwise. The phosphorylated form of species X, with X $\in \{$CheA, CheB, CheY$\}$, is denoted by Xp, while the methylation state of receptor MCP is denoted by MCP^m, for $m = 0,\ldots,4$ (that is, five methylation states are considered).

The reactions describe the following molecular interactions:

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

According to literature, the ternary receptor complex 2MCP^m::2CheW::2CheA is assumed to be stable for the duration of the signal transduction process [22]; moreover, the synthesis and degradation rates of all chemotactic proteins are assumed to occur at a much slower scale than the chemotactic response (hence, the reactions corresponding to these processes have not been included in the model).

A stochastic constant is associated to each reaction: it is needed to evaluate the probability of that reaction to occur when performing stochastic simulations, as explained in Section 3.1. The stochastic constants used for the simulations shown in Section 3.2 are reported in the caption of Table 1 (all values are expressed in sec^{-1}). These values have been partly derived from literature [18], and partly tuned to account for the following biological features [14, 16]: (1) the binding affinity of the ligand is directly proportional to the methylation state of MCPs; (2) the ligand-receptor binding reactions occur at a faster rate with respect to phosphorylation and methylation/demethylation reactions; (3) the methylation and demethylation activities of CheR and CheBp are, respectively, inversely and directly proportional to the methylation state of MCPs; (4) the rate of phosphotransfer from CheA to CheY and CheB depends on the rate of autophosphorylation of CheA. According to these constraints, which set the relative magnitude of some constants with respect to others, the estimation of the unavailable constants has been performed by testing the effect of a range of values for each constant within every module of the model (that is, a group of reactions corresponding to a specific process, such as, e.g., reactions 1-4 that model the formation of the receptor complexes). Within this range, the finally chosen value for each constant is the one that gave a good reproduction of the expected behaviour of the biological subsystem described by that module. Then, every other module has been sequentially added to the previous ones, following the same iterative process, to perform a comprehensive and biologically correct simulation of the whole pathway.

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

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

3 Stochastic simulations of the chemotactic response regulator

In this section, we briefly present the stochastic algorithm used to perform the simulation of the model defined in Section 2.2. Then, we show the results of the dynamics of the chemotactic response regulator protein (the phosphorylated form of CheY) under different conditions of the chemotactic system.

3.1 Tau leaping

The algorithm called *tau leaping* [4] is an approximated and faster version of the seminal Gillespie's stochastic simulation algorithm (SSA) [10]. Both algorithms allow to generate the temporal evolution of chemicals contained inside a well stirred volume, in given and fixed experimental conditions. Chemicals interact with each other by means of given reactions, whose physical and chemical properties are encompassed in a specified stochastic constant associated to each reaction. Reactions are applied according to a probability distribution, that is determined – at each computation step – by the current state of the system (given by the number of molecules of each chemical) and by the value of all reaction constants. SSA and tau leaping share the characteristic that repeated (independent) executions will produce different tempo-

ral dynamics, even starting from the same initial configuration, thus reflecting the inherent noise of the system. The two algorithms differ with respect to the way reactions are applied. In SSA, only *one* reaction can be applied at each step; the reaction that will be actually simulated, and the waiting time before the next reaction will take place, depend on two independent random numbers drawn from the uniform unit interval [0,1]. In tau leaping, instead, *several* reactions can be chosen and executed simultaneously, by the sampling of Poissonian distributions and by choosing an opportune time increment (we refer to [4] for further details). So doing, the computational burden typical of SSA for large systems consisting of many reactions and many molecular species, can be largely reduced. Indeed, with tau leaping it is possible to handle detailed descriptions of many molecular interactions and chemical modifications (e.g., phosphorylation or methylation states of proteins), providing fast and reliable stochastic simulations of mechanistic models of complex biological systems [6]. On the other side, tau leaping is not guaranteed to reproduce the exact behavior of the system, but the accuracy of the simulation can be controlled.

Here we report a sketch of the functioning of tau leaping. We denote by X a well stirred system in thermal equilibrium, consisting of N molecular species $S_1,\ldots,S_N$, which can interact through M chemical reactions $r_1,\ldots,r_M$. Let $X_i(t)$ be the number of molecules of chemical S_i at time t, and $\mathbf{x} = \mathbf{X}(t) \equiv (X_1(t),\ldots,X_N(t))$ the state of the system at time t. The aim of the procedure is to fire several reactions for each time interval $[t,t+\tau)$. In order to find out which reactions will be executed, we have to calculate the probability that a reaction r_j will occur in the next infinitesimal time interval $[t,t+dt)$, starting from the system state $\mathbf{x}$. This probability is given by $a_j(\mathbf{x})dt$, which is the *propensity function* of reaction r_j and is defined as $a_j(\mathbf{x}) = h_j(\mathbf{x}) \cdot c_j$, where $h_j(\mathbf{x})$ is the number of distinct reactant molecules combinations in r_j, and c_j is the stochastic constant associated to r_j.

Given a state $\mathbf{x}$ of the system X, we denote by $K_j(\tau,\mathbf{x},t)$ the exact number of times that a reaction r_j will be fired in the time interval $[t,t+\tau)$, so that $K(\tau,\mathbf{x},t)$ is the exact probability distribution vector (having $K_j(\tau,\mathbf{x},t)$ as elements). For arbitrary values of τ, the computation of the values of $K_j(\tau,\mathbf{x},t)$ is as difficult as resolving the corresponding Chemical Master Equation for that system. On the contrary, if τ is small enough so that the change in the state $\mathbf{x}$ during $[t,t+\tau)$ is so slight that no propensity function will suffer an appreciable change in its value (this is called the *leap condition*), then it is possible to evaluate a good approximation of $K_j(\tau,\mathbf{x},t)$ by using the Poisson random variables with mean and variance $a_j(\mathbf{x}) \cdot \tau$. Hence, starting from the state $\mathbf{x}$ and choosing a τ value that satisfies the leap condition, we can update the state of the system at time $t+\tau$ according to $\mathbf{X}(t+\tau) = \mathbf{x} + \sum_{j=i,\ldots,M} \mathbf{v}_j P_j(a_j(\mathbf{x}),\tau)$, where $P_j(a_j(\mathbf{x}),\tau)$ denotes an independent sample of the Poisson random variable with mean and variance $a_j(\mathbf{x}) \cdot \tau$, and $\mathbf{v}_j \equiv (v_{1j},\ldots,v_{Nj})$ is the state change vector whose element v_{ij} represents the stoichiometric change of the species S_i due to reaction r_j. Summarizing, each iterative step of the algorithm consists of four stages: (1) generate the maximum changes of each species that satisfy the leap condition; (2) compute the mean and variance of the changes of the propensity functions; (3) evaluate the leap value τ exploiting the auxiliary quantities previously computed; (4) toss the reactions to apply; (5) update the system state (see [4] for further details).

The accuracy of this algorithm can be fixed a priori by means of an error control parameter ε ($0 < \varepsilon \leq 1$); for the following simulations, we have used the value $\varepsilon = 0.03$ as suggested in [4]. The algorithm has been implemented in our laboratory [5]; the program is written in C language, compiled under Linux using the GCC compiler. All the simulations have been performed using a Personal Computer with an Intel Core2 CPU (2.66 GHz) running Linux. The mean duration time for one run, for the simulation of the dynamics of CheYp over 3000 sec, is about 4-5 seconds (with the initial values of chemical amounts given in Section 2.2 and the constants reported in Table 1). All the figures reported in Section 3.2, unless otherwise stated, represent the mean value over 50 independent runs of tau leaping, each one executed with the same initial conditions.

3.2 Results of simulations for the dynamics of CheYp

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

We start by reporting in Figure 2, left side, the response of the system to the addition of two consecutive amounts of external ligand: the first stimulus corresponds to a ligand amount of 100 molecules, added to the system at time $t = 3000$ sec and removed at time $t = 6000$ sec, while the second stimulus corresponds to a ligand amount of 500 molecules, added at time $t = 9000$ sec and removed at time $t = 12000$ sec. Note that, since the amount of CheYp is equal to 0 at the beginning of the simulation, its dynamics shows a marked increase which then reaches – due to the counteraction of CheZ, CheR and CheB – a steady state level. Starting from this level, the addition of the ligands has been simulated by changing its amount from 0 to 100 (500, respectively) molecules, thus mimicking the environmental situation where the bacterium encounters a different concentration of attractant molecules. Vice versa, the removal of the ligands has been simulated by putting the value of the ligand back to 0. In the time interval between the addition and the removal of each ligand stimulus, the amount of ligand molecules has been kept at the constant value of 100 and 500, respectively, thus mimicking the presence of an environmental homogenous concentration. This has been done in order to test the adaptation capabilities of the system. In both cases, we can see that the system is able to respond to a step-increase of the ligands by achieving a sharp and fast decrease in CheYp (that is, the negative peaks at time instants $t = 3000$ and $t = 9000$ sec). Immediately after this transient, the amount of CheYp returns to a steady state value, which differs from the prestimulus level only for a few tens of molecules, at most, according to the amount of added ligand. In this phase, the bacterium is returning to the prestimulus switching and thus to the random walk sampling of its surroundings. When the ligand is removed, CheYp shows another transient behavior, corresponding to a sharp and fast increase of its amount, that is in line with experimental observations (see [2, 19]). After this second transient, the amount of CheYp correctly returns to the prestimulus steady state level.

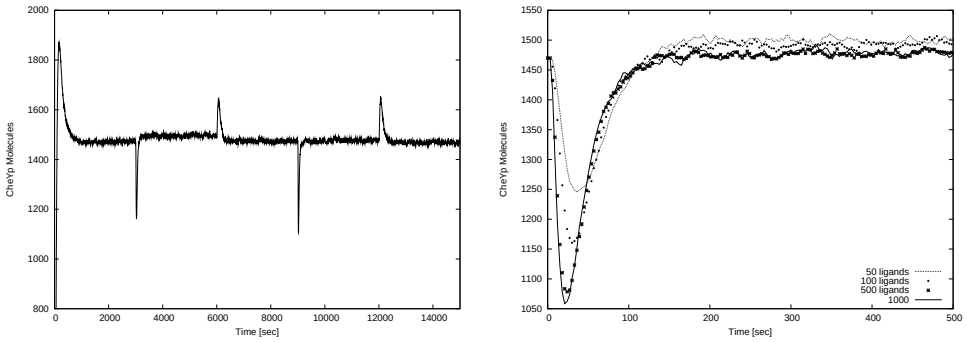


Figure 2: Dynamics of CheYp. Left: adaptation response to two consecutive stimuli. Right: comparison of transient and steady state response to different ligand amounts.

In Figure 2, right side, we compare the transients and steady states reached by CheYp after the addition of distinct ligand amounts. This figure shows that the response magnitude at steady state and the adaptation time of CheYp is only slightly sensitive to the ligand amount, being the relative differences less than a few tens of molecules and less than a few seconds, respectively. The mean values of the steady state of CheYp before the stimulus (SS1), after the ligand addition (SS2) and after the ligand removal (SS3) are reported in Table 2, together with the values of its minimal and maximal values immediately

Table 2: Steady state values and minimal/maximal transient values of CheYp after addition and removal of distinct ligand amounts.

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

after the ligand addition and removal (Min and Max, respectively), for the four ligand amounts (50, 100, 500, 1000 molecules) considered in the right side of Figure 2.

In Figure 3 we show how the dynamics of CheYp changes when CheB is deleted from the system at time $t = 3000$ sec, in both conditions of absence of external ligands (left side) and of presence of 100 molecules of ligand (right side) added at time $t = 3000$ sec. CheB is the methylesterase that, once being phosphorylated by CheA, increases the methylation state of MCPs, thus keeping CheA more active. This, in turn, causes an increase in the amount of CheYp, which is evident from its new steady state level reached after CheB deletion, and also from its less negative transient decrease after ligand addition.

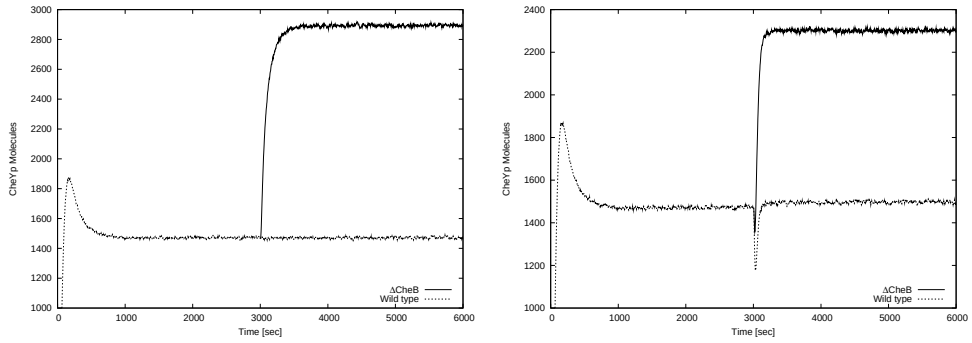


Figure 3: Comparison of dynamics of CheYp in normal condition and after deletion of CheB at $t = 3000$ sec, without ligand (left) and with simultaneous addition of 100 ligand molecules (right).

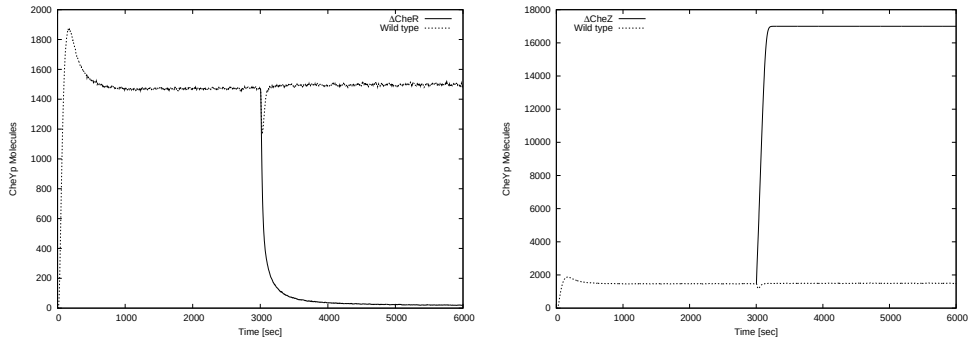


Figure 4: Comparison of dynamics of CheYp in normal condition and after deletion of CheR (left) and CheZ (right) at $t = 3000$ sec, both simulated with a simultaneous addition of 100 ligand molecules.

Similarly, in Figure 4 we show the dynamics of CheYp when either CheR (left side) or CheZ (right

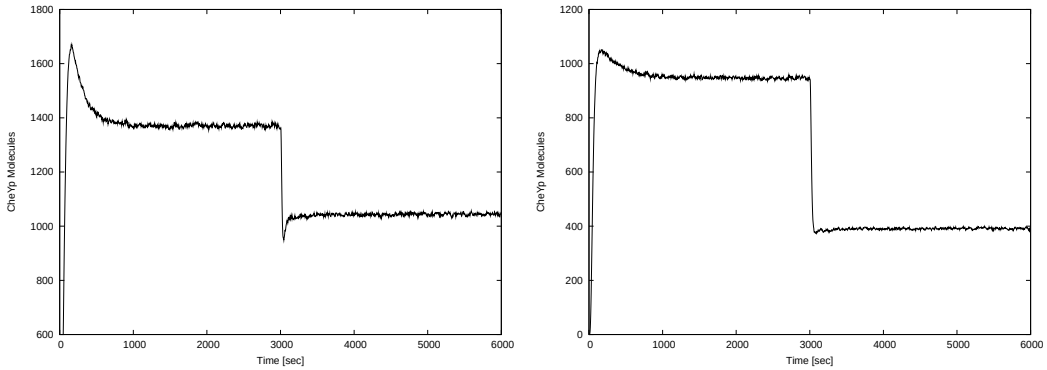


Figure 5: Dynamics of CheYp when only 3 (left) and 2 (right) methylation states are active.

side) are deleted from the system at time $t = 3000$ sec, simultaneously to the addition of 100 ligand molecules (the temporal evolution of CheYp when no ligand is added is basically equivalent). When CheR is deleted, its methyltransferase activity is silenced, the MCPs are no more methylated, and hence the amount of CheYp tends to zero. On the contrary, when CheZ is deleted, all CheY molecules always remain phosphorylated. For the sake of completeness, we have also simulated the dynamics of CheYp when either CheB, CheR or CheZ are deleted from the system since the time instant $t = 0$, in order to have a comparison about the initial temporal evolution of CheYp and the steady state levels it can reach. In these conditions, the model correctly simulates [1, 12, 20] a very low production of CheYp when CheR is deleted, and an increased production (albeit with different magnitudes) when either CheB or CheZ are deleted (data not shown).

Finally, in Figure 5 we compare the dynamics of CheYp in response to the addition of 100 ligand molecules at $t = 3000$ sec, when only 3 (left side) or 2 (right side) methylation states of the receptors are allowed. In practice, this is achieved by initially putting to zero the values of the stochastic constants of methylation and demethylation reactions for levels $m = 4$ and $m = 3$, respectively. In both cases, we see that the system is not able to adapt, as the steady state level of CheYp reached after the addition of the ligand is substantially lower than the steady state when all methylation levels are activated.

The outcome of the stochastic simulations reported in this section validates the model presented in Section 2.2, as the dynamics of CheYp under different conditions of the chemotactic pathway well compare with experimental evidences.

4 The interplay between stochastic fluctuations and the number of bacterial flagella

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

the time required to return to the prestimulus level of CheYp (that is, the transient response immediately after the ligand addition) corresponds to the chemotactic adaptation time – where we expect a much longer and uninterrupted time interval of running motion with respect to the steady state condition.

As explained in Section 2.1, a running motion requires that all flagella are simultaneously synchronized in a CCW rotation – which occurs when CheYp is not interacting with the proteins FliM of the flagellar motors, that is, when its intracellular concentration diminishes with respect to a reference value. To distinguish between the CW and CCW rotations of a *single* flagellum, we assume that the flagellar motor switch is sensitive to a threshold level of CheYp, that is hereby evaluated as the mean value of CheYp at steady state (see also [17], where a similar approach of threshold-crossing mechanism for motor switching was tested, albeit that work considered only a single flagellum and did not propose any investigation on the simultaneous coordination of many flagella). When the amount of CheYp is below this threshold, each flagellum is rotating CCW, while when the amount of CheYp is above the threshold, each flagellum is rotating CW. In what follows, we make a one-to-one correspondence between the behavior of a single flagellum and a temporal evolution of CheYp generated by one run of the tau leaping algorithm, that is, we consider a different and independent stochastic simulation for each and every flagellum (albeit starting from the same initial conditions for the whole system). In other words, we assume that flagella are independent one another – as no molecular interactions between them have been evidenced in bacterial cells. Nonetheless, they all overlook on the same intracellular ambient, that is, they are all subject to the same temporal evolution of CheYp, apart from the stochastic noise existing among independent simulations. In order to determine the synchronization of all flagella that will induce a running motion of the bacterium, we therefore need to identify the time instants when *all* flagella are rotating CCW, that is, to select the time intervals when *all* the temporal evolutions of CheYp are below the fixed threshold.

Formally, we proceed as follows. Let $n = 1,\ldots,10$ be the number of flagella $f_1,\ldots,f_n$ whose influence we want to test, and let s_i, $i = 1,\ldots,n$, be the time series of CheYp (generated by a single tau leaping run) associated to each f_i. For any fixed value of n, the total time of the simulation considered to generate the dynamics of CheYp is the same for all s_i. This simulation time, hereby denoted by Δt_{sim}, is chosen long enough to have a meaningful evaluation of the mean intervals of running and tumbling in the analysis performed below (e.g., $\Delta t_{sim} = 40000, 60000, 120000$ sec for $n = 1, 5, 10$, respectively). The threshold for CheYp is evaluated in the following way: we choose an initial time instant at the steady state level – distant enough from the step decrease of CheYp after ligand addition, i.e. 1000 sec afterward – and then, starting from this instant and till the end of Δt_{sim}, we calculate the mean value $\mu_i = < s_i >$ for each s_i. Then, we define a common threshold μ for all flagella, that is, $\mu = \frac{1}{n}\sum_{i=1,\ldots,n}\mu_i$. This threshold is then considered as the reference value also for the portion of the CheYp dynamics corresponding to the transient decrease after ligand addition. In Figure 6, top panel on the left side, we show a part of Δt_{sim} over a single simulation of CheYp, where both the initial transient response and the stochastic fluctuations around the threshold are evidenced. For all the results discussed below, the different values of μ have been found to be approximatively equal to 1480 molecules.

The next step consists in detecting, for each f_i, the time intervals during which the amount of CheYp remains below μ, each one of these intervals corresponding to a CCW rotation time interval of that flagellum. Namely, for each s_i we identify the time intervals $\Delta t_{true} \subseteq \Delta t_{sim}$ such that $\Delta t_{true} = \{t \in \Delta t_{sim} \mid s_i(t) - \mu \leq 0\}$. Note that this simple mechanism of single threshold-crossing could be extended to consider more complex situations – e.g., a double threshold-crossing mode can be assumed – whereby one simply asks for analogous conditions to be satisfied. Similarly, for each s_i we can locate the complementary time intervals $\Delta t_{false} \subseteq \Delta t_{sim}$ such that $\Delta t_{false} = \{t \in \Delta t_{sim} \mid s_i(t) - \mu > 0\}$; these intervals correspond to the time that each flagellum f_i spends in a CW rotation. Stated in other terms, we can associate to

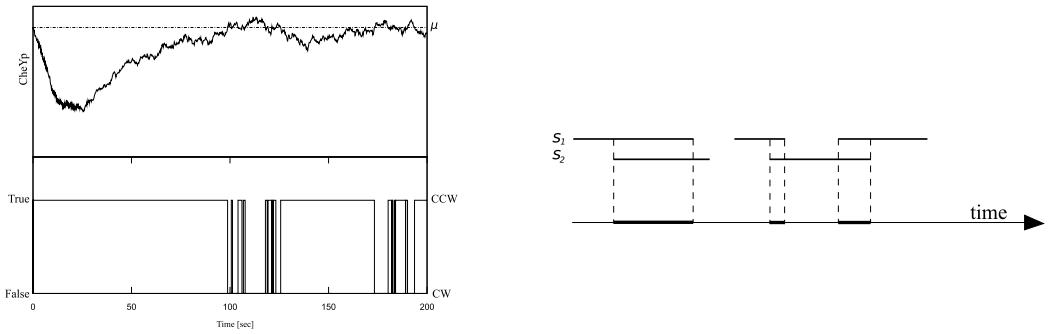


Figure 6: Threshold-crossing intervals in stochastic fluctuations of CheYp (left) and synchronization of running motion between 2 flagella (right).

each s_i a function $CCW_{s_i} : \Delta t_{sim} \to \{true, false\}$ defined as:

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

In Figure 6, bottom panel on the left side, we show the values of this function for the CheYp simulation given in the upper panel. As it can be seen at a glance, the transient response after ligand addition (when the amount of CheYp is initially below μ) corresponds to a longer and uninterrupted interval of CCW rotation of that flagellum.

Once that the set of all Δt_{true} intervals – or, equivalently, of all functions CCW_{s_i} – have been found out for each flagellum, we start the process of synchronization for any given number n of flagella. To this aim, let us define $\mathscr{T}^n_{sync} = \{t \in \Delta t_{sim} \mid CCW_{s_i}(t) = true \text{ for all } i = 1, \ldots, n\}$. $\mathscr{T}^n_{sync}$ is the set of all times during which *all* time series s_i are below the threshold μ, that is, the time intervals during which *all* flagella are rotating CCW. More precisely, we identify these intervals as the running motion of the bacterium, i.e. $\mathscr{T}^n_{sync}$ corresponds to the time of directional swimming – when all flagella are coordinated in a bundle. As an example, in Figure 6, right side, we represent the functions $CCW_{s_i}(t) = true$ for $i = 1, 2$, and the corresponding set $\mathscr{T}^n_{sync}$, $n = 2$. The complementary set, $\mathscr{T}^n_{unsync} = \Delta t_{sim} \setminus \mathscr{T}^n_{sync}$, corresponds instead to tumbling motion – when at least one flagellum (over the set of n flagella considered time by time) is rotating CW. Namely, $\mathscr{T}^n_{unsync} = \{t \in \Delta t_{sim} \mid \text{ there exists } i = 1, \ldots, n \text{ such that } CCW_{s_i}(t) = false\}$.

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

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

The results for $\langle \Delta t_{run} \rangle$ are reported in Figure 7, left side, where we can see that the mean time intervals of running motion are very short, and their values decrease in a (qualitative) exponential way as

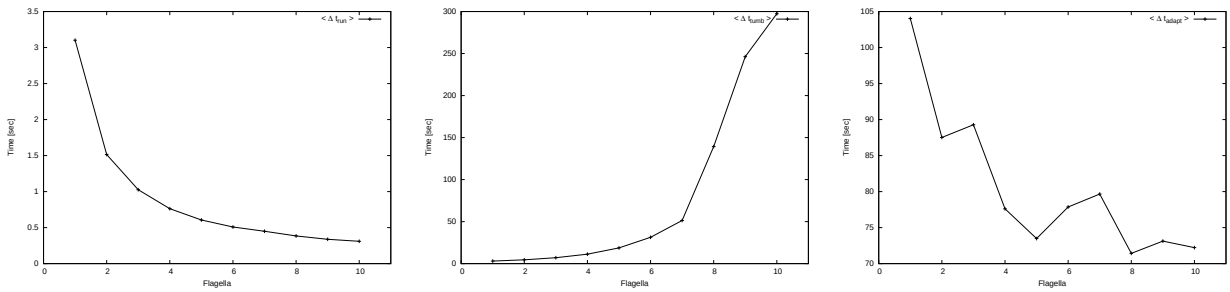


Figure 7: Variation of mean time values of running (left) and tumbling motions (middle), and of adaptation time (right), with respect to the number of flagella.

Table 3: Values of mean time intervals for running, tumbling and adaptation.

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

the number n of flagella increases, as expected. Similarly, the results for $\langle \Delta t_{tumb} \rangle$ evidence a (qualitative) exponential increase with respect to n, as reported in Figure 7, middle part. As reference, the precise values of the mean running and tumbling time intervals are given in Table 3, together with their ratio, for three values of n. The running-to-tumbling ratio, which decreases as n increases, highlights the relevance of the number of flagella and the necessity of their synchronization with respect to the chemotactic behavior of the bacterium. That is, we see that for $n = 1$ the time spent in running or tumbling motions is approximatively equivalent, but if coordination among many flagella ($n = 10$) has to take place, then the running motions are highly reduced with respect to tumbling motions, which is in agreement with biological expectations.

The results for $\langle \Delta t_{adapt} \rangle$ are reported in Figure 7, right side, and in Table 3. In this case, it is not possible to recognize a simple function for the curve progress, and we see that the variation of the time intervals is within a range of a few tens of seconds. Once more, this result seems to be in agreement with biological expectations, as the response of the bacterium to an environmental change (i.e. the addition or removal of ligands) should not be strictly dependent on the number of flagella that are present on its surface, otherwise the chemotactic pathway would not guarantee an appropriate adaptation mechanism, independently from the variation of the number of flagella among distinct individuals in a population of cells.

5 Discussion

In this paper we have investigated the possible influence of stochastic fluctuations of the chemotactic protein CheYp on the running motion of bacterial cells, with respect to an increasing number of flagella in the individual bacterium. To this aim, we have defined a procedure to identify the synchronization of CCW rotations of each and every flagella, and then we have compared the mean time intervals of running and tumbling motions of the cell, as well as of adaptation times to ligand addition, according to the different numbers of flagella. We have shown that, on the one hand, the running-to-tumbling ra-

tio highlights the relevance of the number of flagella, and the necessity of their synchronization with respect to the chemotactic behavior of the bacterium. On the other hand, the adaptation time does not seem to be strongly influenced by the varying number of flagella in distinct individual cells. These results have been obtained by performing stochastic simulations of a very detailed mechanistic model of the bacterial chemotaxis pathway, that takes into account all proteins, and their respective interactions, involved in both signaling and response. All post-translational modifications of proteins, such as methylation and phosphorylation, have been considered because of their relevant roles in the feedback control mechanisms governing this pathway. In particular, by exploiting the efficiency of tau leaping algorithm, we have investigated the dynamics of the pivotal protein involved in chemotaxis, CheYp, under different conditions, such as the deletion of other chemotactic proteins, the addition of distinct amounts of external ligand, the effect of different methylation states of the receptors.

Concerning the analysis of the interplay between CheYp fluctuations and the number of flagella, other relevant biological aspects of chemotaxis, that stand downstream of the signaling process, might represent valuable points to be considered for future research. For instance, it is known that each flagellar motor switch-complex is constituted by two principal components: a group of proteins, called FliM, FliN, FliG (assembled in a ring), and the torque-generating complexes, called MotA and MotB. In *E. coli,* a typical flagellar ring contains 34 copies of FliM, each of which can bind one copy of CheYp. In [8] it is suggested that binding of CheYp to FliM modifies the displacement of protein FliG, which directly interacts with the Mot complexes and therefore modulates the switch state of the flagellum. Moreover, flagellar motor switching has been found to be highly sensitive to the concentration of CheYp (having a Hill coefficient ≈ 10), though the binding of CheYp to FliM has a low level of cooperativity (Hill coefficient ≈ 1). In [8], the hypothesis that CheYp can interact more favorably with the FliM displaced in the CW orientation, than those in the CCW orientation, is put forward. In [20], in addition, the following mechanism is considered for the control of flagellar motor by means of CheYp: the number of CheYp molecules bound to FliM determines the probability of CW or CCW rotation, while the switch is thrown by thermal fluctuations. In other words, CheYp only changes the stabilities of the two rotational states, by shifting the energy level of CCW-state up and of CW-state down, by a magnitude that is directly proportional to the number of bound molecules. Therefore, interesting questions related to stochastic fluctuations of CheYp, that might be coupled with the investigation on the number of cellular flagella, are: how many FliM proteins at each flagellar switch have to be occupied by CheYp in order to generate the CW rotation of each flagellum and of the bacterium [3]? What is the corresponding probability of throwing the reversal switch? Can a double-threshold crossing mechanism [3] be more suitable to effectively detect the CW and CCW rotational states?

Nonetheless, other related features might be relevant points for a further extension of this work. For instance, the gradient of CheYp that can be present inside the cytoplasm – due to the diffusion from the area of its phosphorylation (close to chemotactic receptors) to the area of its activity (around the flagellar motors) – can be a significant aspect in chemotaxis, together with the localization of CheZ (that controls the dephosphorylation of CheYp) and of the flagella around the cell [15]. With respect to this matter, how are diffusion processes related to the interactions between CheYp and the flagellar motors? And how does diffusion intervene on the chemotactic response?

We believe that the definition of detailed mechanistic models, like the one proposed in this paper for chemotaxis, coupled to the use of efficient procedures for the analysis of stochastic processes in individual cells, can be a good benchmark to investigate the combined roles of many biological factors interplaying within a common system. With this perspective, the development of formal methods specifically devised for the analysis of properties (e.g., synchronization) of stochastic systems represents indeed a hot topic research in biological modeling.

References

[1] U. Alon, M. G. Surette, N. Barkai & S. Leibler (1999): *Robustness in bacterial chemotaxis. Nature* 397(6715), pp. 168–171. Available at http://dx.doi.org/10.1038/16483.

[2] N. Barkai & S. Leibler (1997): *Robustness in simple biochemical networks. Nature* 387(6636), pp. 913–917. Available at http://dx.doi.org/10.1038/43199.

[3] A. Bren & M. Eisenbach (2001): *Changing the direction of flagellar rotation in bacteria by modulating the ratio between the rotational states of the switch protein FliM. J. Mol. Biol.* 312(4), pp. 699–709. Available at http://dx.doi.org/10.1006/jmbi.2001.4992.

[4] Y. Cao, D. T. Gillespie & L. R. Petzold (2006): *Efficient step size selection for the tau-leaping simulation method. J. Chem. Phys.* 124(4), p. 044109. Available at http://dx.doi.org/10.1063/1.2159468.

[5] P. Cazzaniga, D. Pescini, D. Besozzi & G. Mauri (2006): *Tau leaping stochastic simulation method in P systems*. In: H. J. Hoogeboom, G. Păun, G. Rozenberg & A. Salomaa, editors: *7th International Workshop on Membrane Computing WMC 2006*, 4361. LNCS, pp. 298–313. Available at http://dx.doi.org/10.1007/11963516_19.

[6] P. Cazzaniga, D. Pescini, D. Besozzi, G. Mauri, S. Colombo & E. Martegani (2008): *Modeling and stochastic simulation of the Ras/cAMP/PKA pathway in the yeast Saccharomyces cerevisiae evidences a key regulatory function for intracellular guanine nucleotides pools. J. Biotechnol.* 133(3), pp. 377–385. Available at http://dx.doi.org/10.1016/j.jbiotec.2007.09.019.

[7] M. Dugo (2009): *Modellazione stocastica della chemiotassi batterica*. Bachelor thesis, Università degli Studi di Milano.

[8] C. M. Dyer, A. S. Vartanian, H. Zhou & F. W. Dahlquist (2009): *A Molecular Mechanism of Bacterial Flagellar Motor Switching. J. Mol. Biol.* 388, pp. 71–84. Available at http://dx.doi.org/10.1016/j.jmb.2009.02.004.

[9] M. B. Elowitz, A. J. Levine, E. D. Siggia & P. S. Swain (2002): *Stochastic gene expression in a single cell. Science* 297, pp. 1183–1186. Available at http://dx.doi.org/10.1126/science.1070919.

[10] D. T. Gillespie (1977): *Exact stochastic simulation of coupled chemical reactions. Journ. Phys. Chem.* 81(25), pp. 2340–2361. Available at http://dx.doi.org/10.1021/j100540a008.

[11] M. S. Jurica & B. L. Stoddard (1998): *Mind your B's and R's: bacterial chemotaxis, signal transduction and protein recognition. Structure* 6(7), pp. 809–813. Available at http://www.ncbi.nlm.nih.gov/pubmed/9687374.

[12] C. Kim, M. Jackson, R. Lux & S. Khan (2001): *Determinants of chemotactic signal amplification in Escherichia coli. J. Mol. Biol.* 307, pp. 119–135. Available at http://dx.doi.org/10.1006/jmbi.2000.4389.

[13] M. D. Levin, C. J. Morton-Firth, W. N. Abouhamad, R. B. Bourret & D. Bray (1998): *Origins of individual swimming behavior in bacteria. Biophys. J.* 74(1), pp. 175–181. Available at http://dx.doi.org/10.1016/S0006-3495(98)77777-X.

[14] G. Li & R. M. Weis (2000): *Covalent modification regulates ligand binding to receptor complexes in the chemosensory system of Escherichia coli. Cell* 100(3), pp. 357–365. Available at http://view.ncbi.nlm.nih.gov/pubmed/10676817.

[15] K. Lipkow, S. S. Andrews & D. Bray (2005): *Simulated diffusion of phosphorylated CheY through the cytoplasm of Escherichia coli. J. Bacteriol.* 187(1), pp. 45–53. Available at http://dx.doi.org/10.1128/JB.187.1.45-53.2005.

[16] B. A. Mello & Y. Tu (2003): *Perfect and near-perfect adaptation in a model of bacterial chemotaxis. Biophys. J.* 84(5), pp. 2943–2956. Available at http://dx.doi.org/10.1016/S0006-3495(03)70021-6.

[17] C. J. Morton-Firth & D. Bray (1998): *Predicting temporal fluctuations in an intracellular signalling pathway. J. Theor. Biol.* 192(1), pp. 117–128. Available at http://dx.doi.org/10.1006/jtbi.1997.0651.

[18] C. J. Morton-Firth, T. S. Shimizu & D. Bray (1999): *A free-energy-based stochastic simulation of the Tar receptor complex. J. Mol. Biol.* 286(4), pp. 1059–1074. Available at http://dx.doi.org/10.1006/jmbi.

1999.2535.

[19] C. V. Rao, J. R. Kirby & A. P. Arkin (2004): *Design and diversity in bacterial chemotaxis: a comparative study in Escherichia coli and Bacillus subtilis. PLoS Biol.* 2(2), p. E49. Available at http://dx.doi.org/10.1371/journal.pbio.0020049.

[20] B. E. Scharf, K. A. Fahrner, L. Turner & H. C. Berg (1998): *Control of direction of flagellar rotation in bacterial chemotaxis. Proc. Natl. Acad. Sci. U S A* 95(1), pp. 201–206. Available at http://www.pnas.org/content/95/1/201.abstract.

[21] T. S. Shimizu & D. Bray (2002): *Modelling the bacterial chemotaxis receptor complex. Novartis Found. Symp.* 247, pp. 162–77; discussion 177–81, 198–206, 244–52. Available at http://www.pdn.cam.ac.uk/groups/comp-cell/Papers/Shimizu02b.pdf.

[22] V. Sourjik (2004): *Receptor clustering and signal processing in E. coli chemotaxis. Trends Microbiol.* 12(12), pp. 569–576. Available at http://dx.doi.org/10.1016/j.tim.2004.10.003.

[23] P. A. Spiro, J. S. Parkinson & H. G. Othmer (1997): *A model of excitation and adaptation in bacterial chemotaxis. Proc. Natl. Acad. Sci. U S A* 94(14), pp. 7263–7268. Available at http://www.pnas.org/content/94/14/7263.abstract.

[24] T. E. Turner, S. Schnell & K. Burrage (2004): *Stochastic approaches for modelling in vivo reactions. Computational Biology and Chemistry* 28, pp. 165–178. Available at http://dx.doi.org/10.1016/j.compbiolchem.2004.05.001.

[25] G. H. Wadhams & J. P. Armitage (2004): *Making sense of it all: bacterial chemotaxis. Nat. Rev. Mol. Cell. Biol.* 5(12), pp. 1024–1037. Available at http://dx.doi.org/10.1038/nrm1524.

Hybrid Semantics of Stochastic Programs with Dynamic Reconfiguration

Luca Bortolussi
Dept. of Mathematics and Informatics,
University of Trieste, Italy.
luca@dmi.units.it

Alberto Policriti
Dept. of Mathematics and Informatics,
University of Udine, Italy.
Istituto di Genomica Applicata, Udine, Italy.
policriti@dimi.uniud.it

We begin by reviewing a technique to approximate the dynamics of stochastic programs—written in a stochastic process algebra—by a hybrid system, suitable to capture a mixed discrete/continuous evolution. In a nutshell, the discrete dynamics is kept stochastic while the continuous evolution is given in terms of ODEs, and the overall technique, therefore, naturally associates a Piecewise Deterministic Markov Process with a stochastic program.

The specific contribution in this work consists in an increase of the flexibility of the translation scheme, obtained by allowing a *dynamic* reconfiguration of the degree of discreteness/continuity of the semantics.

We also discuss the relationships of this approach with other hybrid simulation strategies for biochemical systems.

1 Introduction

Models in systems biology tend to cluster around two families of mathematical tools: differential equations and stochastic processes. Even though, physically speaking, stochastic models have firmer grounds [14, 15], their computational analysis is much more costly than that of their differential counterpart. In any case, ODE-based descriptions of biological systems are often valuable and provide deep insights. Indeed, it is known that, limiting to mass action models, ODE's are an approximation of the average of stochastic models, and the differences between the two vanish in the thermodynamic limit [13] (i.e. when populations and system's size go to infinity). Recently, there have been many attempts to mix these two techniques, at least as far as simulation of biological systems is concerned, resulting in several hybrid simulation algorithms [32, 25]. Hybrid dynamical systems have also been a hot topic in the last two decades, with much research work spanning across the boundary between computer science and engineering control. The best known model among hybrid dynamical systems are *hybrid automata* [21]. Stochastic extensions of such concept are also receiving recently much attention [10], although stochastic hybrid systems have a somewhat longer tradition [12]. In both cases, most of the interest is in the development of automated reasoning tools rather than in simulation.

It is widely recognized that Computational Systems Biology can highly benefit from modeling approaches embodying *some* stochastic ingredient. A very popular line along which such incorporation is realized, is based on the use of stochastic process algebras [27, 11], which are proposed as front-end languages to (automatically) generate mathematical models, usually Continuous Time Markov Chains(CTMC), see [32]. Recently, such process algebra based languages have also been endowed with semantics based on ODE [22], which increase the flexibility of such tools.

Many proposals of hybrid simulation algorithms for systems of biochemical reactions have been put forward [19, 24, 20, 28, 23, 16, 2]. Their salient feature is a description of one part of the system as

R.J.Back, I.Petre, E. de Vink (Eds.): Computational Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 63–76, doi:10.4204/EPTCS.6.5

continuous, while keeping the other discrete and stochastic. The basic idea is to find the best trade off between accuracy and computational efficiency (stochastic simulations are much more expensive than ODE simulation).

In this paper we continue a programme which aims to increase even more the flexibility of stochastic process algebras by providing them with a very general semantics based on (stochastic) hybrid systems, encompassing CTMC and ODE as special cases. Such an approach is motivated not only by the gain in flexibility, but also by the goal of exploiting, in a systematic manner, automated reasoning tools to provide as much information as possible from a given model. Our stochastic process algebra of choice is stochastic Concurrent Constraint Programming (**sCCP**) [6], an extension of CCP [29] in the stochastic setting. In addition to the standard CTMC-based semantics, we have also provided sCCP with an ODE-based semantics [4] and with an hybrid automata based semantics. Moreover, hybrid semantics has been proposed both with a fixed or user-defined amount of continuously approximated components (see [8, 9]).

In this paper we extend our work by introducing a semantics based on *Stochastic* Hybrid Automata, thereby guaranteeing the possibility of parameterizing the degree of continuity introduced in the model. The approach allows also a *dynamic* reconfiguration of such degree, in accordance to properties of the current state of the system. This allows the description in a formal setting of different hybrid simulation strategies, opening the way for their use in the context of process algebra modelling.

We will start our presentation by introducing, in Section 2, a high level description of the target stochastic hybrid systems, suitable to be easily mapped to the well-established formalism of Piecewise Deterministic Markov Processes (see supplementary material [1]). The formalism introduced in Section 2, called Transition-Driven Stochastic Hybrid Automata (TDSHA), will act as the intermediate layer in the definition of the stochastic hybrid semantics of **sCCP**. Section 3 briefly introduces the **sCCP** language, while Section 4 presents the mapping from **sCCP** to TDSHA. Collections of TDSHAs can be organized in a lattice, whose definition and basic properties are presented in Section 4.1. Finally, in Section 5 we introduce the dynamic reconfiguration mechanism, briefly discussing also how to render, in such reconfigurations, partition strategies developed for hybrid simulation algorithms.

2 Transition-driven Stochastic Hybrid Automata

We define here a stochastic variant of *Transition-Driven Hybrid Automata*, introduced in [9] as an intermediate layer to map **sCCP** into hybrid automata. The emphasis is on *transitions* which, as always in hybrid automata, can be either discrete (corresponding to jumps) or continuous (representing flows acting on system's variables). The stochastic variant defined below contains two kind of discrete transitions: instantaneous—as in [9]—and stochastic, which happen with an hazard given by a rate function.

Definition 2.1. A Transition-Driven Stochastic Hybrid Automaton (TDSHA) is a tuple $\mathscr{T} = (Q, \mathbf{X}, \mathfrak{TC}, \mathfrak{TD}, \mathfrak{TS}, init)$, where:

- Q is a finite set of *control modes*.
- $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of real valued *system's variables*[1].
- $\mathfrak{TC}$ is the set of *continuous transitions or flows*, whose elements τ are triples $(q, stoich, rate)$, where: $q \in Q$ is a mode, *stoich* is a vector of size $|\mathbf{X}|$, and $rate : \mathbb{R}^n \to \mathbb{R}$ is a (sufficiently smooth) function. The elements of a triple τ are indicated by $\mathbf{cmode}[\tau]$, $\mathbf{stoich}[\tau]$, and $\mathbf{rate}[\tau]$, respectively.

[1] Notation: the time derivative of X_j is denoted by $\dot{X}_j$, while the value of X_j after a change of mode is indicated by X'_j

- $\mathfrak{TD}$ is the set of *instantaneous transitions*, whose elements δ are tuples of the form $(q_1, q_2, priority, guard, reset)$, where: q_1 is the *exit-mode*, q_2 is the *enter-mode*, $priority : \mathbb{R}^n \to \mathbb{R}^+$ is a weight function used to resolve non-determinism between two or more active transitions. Moreover, *guard* is a quantifier-free first-order formula with free variables in $\mathbf{X}$, representing the *closed set* $G_\delta = \{\mathbf{x} \in \mathbb{R}^n \mid guard[\mathbf{x}]\}$ in which thew transition is active, and *reset* is a deterministic update of the form $\mathbf{X}' = f(\mathbf{X})$.[2] The elements of a tuple δ are indicated by $\mathbf{e_1}[\delta]$, $\mathbf{e_2}[\delta]$, $\mathbf{priority}[\delta]$, $\mathbf{guard}[\delta]$, and $\mathbf{reset}[\delta]$, respectively.
- $\mathfrak{TS}$ is the set of *stochastic transitions*, whose elements η are tuples of the form $\eta = (q_1, q_2, guard, reset, rate)$, where q_1, q_2, *guard*, and *reset* are as for transitions in $\mathfrak{TD}$, while $rate : \mathbb{R}^n \to \mathbb{R}^+$ is the rate function giving the hazard of taking transition η. Such function is referred to by $\mathbf{rate}[\eta]$.
- *init* is a point giving the initial state of the system.

A TDSHA has three types of transitions. Continuous transitions represent flows and, for each $\tau \in \mathfrak{TC}$, $\mathbf{stoich}[\tau]$ and $\mathbf{rate}[\tau]$ give the *magnitude* and the *form* of the flow of τ on each variable $X \in \mathbf{X}$, respectively (see below). Instantaneous transitions represent actions happening immediately when their guard becomes true. Finally, stochastic transitions happen at a specific rate. Both instantaneous and stochastic transitions can change system variables according to a specific reset function, depending on the variables'value at the point in time at which the jump occurs.

Remark 2.1. Both priority and rates introduced in Definition 2.1 make TDSHA stochastic. Priorities define, at each point, a discrete distribution of a random variable choosing among enabled instantaneous transitions. Rates, on the other hand, define a random race in continuous time, giving the delay for the next spontaneous jump.

Product of TDSHA. Given two TDSHA $\mathscr{T}_1 = (Q_1, \mathbf{X_1}, \mathfrak{TC}_1, \mathfrak{TD}_1, \mathfrak{TS}_1, init_1)$ and $\mathscr{T}_2 = (Q_2, \mathbf{X_2}, \mathfrak{TC}_2, \mathfrak{TD}_2, \mathfrak{TS}_2, init_2)$, the product TDSHA $\mathscr{T} = \mathscr{T}_1 \otimes \mathscr{T}_2$ can be defined in a simple way, along the path outlined in [9]. Essentially, the discrete states'space of the product automaton is $Q_1 \times Q_2$, while transitions from state (q_1, q_2) are all those issuing from q_1 *or* q_2. Instantaneous or stochastic transitions of $\mathscr{T}_1$ going from state q_1 to state q_1', will go from a state (q_1, q_2) to (q_1', q_2) for each $q_2 \in Q_2$. Symmetrically for transitions of $\mathscr{T}_2$.

Dynamics of TDSHA. In order to formally define the dynamical evolution of TDSHA, we can map them into a well-studied model of Stochastic Hybrid Automata, namely Piecewise Deterministic Markov Processes [12]. In this sense, TDSHA are related to communicating PDMP [31], as they can also be seen as a compositional formalism to model PDMP. Due to space constraints, we just sketch here an informal description of PDMP. The interested reader can find a more formal treatment of PDMP and of their relation with TDSHA in the supplementary material [1].

Basically, PDMP are stochastic processes whose *state space* is given by a finite collection of *discrete modes* and by a set of *real-valued variables*. Within each mode, the continuous variables evolve following the solution of a set of *mode-specific ODE's*. While in a mode, variables must stay within the *allowed region*. If they touch the boundary of the allowed region, a *forced discrete transition* is taken, and the system may change mode and/or reset the value of the variables. Moreover, the system is subject to the happening *discrete stochastic events*, governed by an hazard rate that is function of the discrete

[2] Even though there is no real additional difficulty in considering stochastic resets—i.e. in assuming *reset* to be a transition measure—we decided to avoid such move for the sake of simplicity.

mode and of continuous variables. Also stochastic transitions trigger a reset of the state of the system. The main points of the mapping from TDSHA to PDMP are the following.

- Within each discrete mode $q \in Q$, the system follows the solution of a system of ODE, constructed combining the effects of the continuous transitions τ acting on mode q. Essentially, the ODE for variable X_i is obtained by adding up the rate of all such τ times the i-th component of the vector $\mathbf{stoich}[\tau]$:
$$\dot{\mathbf{X}} = \sum_{\tau|\ |\mathbf{cmode}[\tau]=q} \mathbf{stoich}[\tau]\mathbf{rate}[\tau] \quad \text{in mode } q \in \mathbb{Q}.$$
- Two kinds of discrete jumps are possible: stochastic transitions are fired according to their rate, while instantaneous transitions are fired as soon as their guard becomes true. In both cases, the state of the system is reset according to the policy specified by **reset**. Choice among several active stochastic or instantaneous transitions is resolved probabilistically according to their rate or priority, see Remark 2.1.
- A trace of the system is therefore a sequence of (random) jumps interleaved by periods of continuous evolution.

3 Stochastic Concurrent Constraint Programming

In this section we briefly present (a simplified version of) stochastic Concurrent Constraint Programming (**sCCP** [3], a stochastic extension of CCP [30]), as it seems to be sufficiently expressive, compact, and especially easy to manipulate for our purposes[3]. In the following we just sketch the basic notions and the concepts needed in the rest of the paper. More details on the language can be found in [3, 6].

Definition 3.1. A **sCCP** program is a tuple $\mathscr{A} = (A, \mathscr{D}, \mathbf{X}, init(\mathbf{X}))$, where

1. The *initial network of agents* A and the *set of definitions* $\mathscr{D}$ are given by the following grammar:
$$\mathscr{D} = \emptyset \mid \mathscr{D} \cup \mathscr{D} \mid \{C \stackrel{\text{def}}{=} M\}$$
$$\pi = [g(\mathbf{X}) \to u(\mathbf{X}, \mathbf{X}')]_{\lambda(\mathbf{X})} \quad M = \pi.C \mid M + M \quad A = M \mid A \parallel A$$
2. $\mathbf{X}$ is the set of variables of the store (with global scope);
3. $init(\mathbf{X})$ is a predicate on $\mathbf{X}$ of the form $\mathbf{X} = \mathbf{x_0}$, assigning an *initial value* to store variables.

In the previous definition, basic actions are *guarded updates* of (some of the) variables: $g(\mathbf{X})$ is a quantifier-free first order formula whose atoms are inequality predicates on variables $\mathbf{X}$ and $u(\mathbf{X}, \mathbf{X}')$ is a predicate on $\mathbf{X}, \mathbf{X}'$ of the form $\mathbf{X}' = f(\mathbf{X})$ ($\mathbf{X}'$ denotes variables of $\mathbf{X}$ after the update), for some function $f : \mathbb{R}^n \to \mathbb{R}^n$. Each such action has a *stochastic duration*, specified by associating an exponentially distributed random variable to actions, whose rate depends on the state of the system through a function $\lambda : \mathbf{X} \to \mathbb{R}^+$.

Example 3.1. We will illustrate the notions introduced in the paper by means of an example coming from biological systems. Specifically, we consider a simple model of a (procaryotic) genetic regulatory network with a single gene, expressing a protein acting, after dimerization, as a repressor of its own

[3] There are other probabilistic extensions of CCP studied in literature, like [26, 18, 17]. [26] provides CCP with a semantics based on discrete time Markov Chains, while in [18, 17] the stochastic ingredient is introduced by extending the store with random variables and adding a primitive for sampling. These approaches, however, are not suited for our purposes, as we need a model in which events happen probabilistically in continuous-time, as customary in biochemical modeling.

production. We assume a cooperative repression: two dimers are required to bind to the promoter region of the gene. The **sCCP** model is given by $\mathscr{A} = (A, \mathscr{D}, \mathbf{X}, init)$, where the variables are $\mathbf{X} = \{X_p, X_{p2}\}$, storing the quantity of the protein p and of its dimer $p2$ and the components in $\mathscr{D}$ are (* stands for true):

$$
\begin{array}{lcl}
\texttt{gene}_0 & \stackrel{\text{def}}{=} & [* \to X_p' = X_p + 1]_{k_{p1}} . \texttt{gene}_0 \ \texttt{+} \ [X_{p2} > 0 \to *]_{k_{p1}X_{p2}} . \texttt{gene}_1 \\
\texttt{gene}_1 & \stackrel{\text{def}}{=} & [* \to X_p' = X_p + 1]_{k_{p2}} . \texttt{gene}_1 \ \texttt{+} \ [X_{p2} > 0 \to *]_{k_{p2}X_{p2}} . \texttt{gene}_2 \ \texttt{+} \\
 & & [* \to *]_{k_{u1}} . \texttt{gene}_0 \\
\texttt{gene}_2 & \stackrel{\text{def}}{=} & [* \to *]_{k_{u2}} . \texttt{gene}_1 \\
\texttt{deg} & \stackrel{\text{def}}{=} & [* \to X_p' = X_p - 1]_{k_d X_p} . \texttt{deg} \\
\texttt{dimer} & \stackrel{\text{def}}{=} & [* \to X_p' = X_p - 2 \wedge X_{p2}' = X_{p2} + 1]_{k_x X_p (X_p - 1)/2} . \texttt{dimer} \ \texttt{+} \\
 & & [* \to X_p' = X_p + 2 \wedge X_{p2}' = X_{p2} - 1]_{k_{-x} X_{p2}} . \texttt{dimer}
\end{array}
$$

The initial network A is $\texttt{gene}_0 \parallel \texttt{deg} \parallel \texttt{dimer}$ with initial values of the store variables are given by

$$init(X_p, X_{p2}) = (X_p = 0) \wedge (X_{p2} = 0).$$

Notice: there is no need to introduce agents for proteins or dimers, as the quantity of these objects needs only to be measured by stream variables. The repression mechanism is represented by a gene unable of expressing a protein whenever in state $\texttt{gene}_2$. We did not decrement X_{p_2} before entering states $\texttt{gene}_1$ and $\texttt{gene}_2$ as we assume repression mechanism not requiring a binding of the dimer (inhibition by bumping).

Remark 3.1. The pros and cons of using **sCCP** as a modeling language for biological systems are discussed in detail in [6]. Basically, **sCCP** combines on one side the logical simplicity of process algebras and on the other side the computational power of constraints. As a matter of fact, the constraint store can be more general than that used in this paper, whereby more complex information (like spatiality) can be managed just by a simple programming activity. Further work is needed, however, to export the techniques developed here to a more general version of the store.

All agents definable in **sCCP**, i.e. all agents $C \stackrel{\text{def}}{=} M \in \mathscr{D}$,[4] are *sequential*, i.e. they do not contain any occurrence of the parallel operator, whose usage is restricted at the upper level of the network.
sCCP sequential agents can be seen as automata synchronizing on store variables and they can be conveniently represented as labeled graphs, called *Reduced Transition Systems* (RTS) (see [5]).

The steps to obtain an object suitable to our subsequent treatment are the following:

1. Define the collection of all possible states—the *derivative set* $Der(C)$—and actions—$action(C)$—of any sequential agent appearing in a **sCCP** program.
2. Restrict to **sCCP** *simple programs*, i.e. programs without multiple copies of the same agent running in parallel at the same time. Formally, it is required that the derivative sets of any two agents in parallel in the initial network are disjoint. This is only an apparent restriction, cf. [9] for a more detailed discussion.
3. Introduce the following multi-graph[5] $R\text{TS}(C) = (S(C), E(C), \ell)$:
   - $S(C) = Der(C)$,
   - $E(C) = \{(exit(\pi), enter(\pi)) \mid \pi \in action(C)\}$,

[4] In the following, with a slight abuse of notation, we sometimes write $C \in \mathscr{D}$ for $C \stackrel{\text{def}}{=} M \in \mathscr{D}$.

[5] $exit(\pi), enter(\pi), guard(\pi), update(\pi), rate(\pi)$ give the executing agent, the target agent, the guard, the update and the rate of an action π, respectively.

- $\ell(e) = (guard(\pi), update(\pi), rate(\pi))$, where π is the action defining $e \in E(C)$.

In Figure 1, we show the RTS for the agent $\texttt{gene}_0$, defined in Example 3.1.

4. Introduce the notion of *extended* **sCCP** program

$$\mathscr{A}^+ = (A^+, \mathscr{D}^+, \mathbf{X} \cup \{P_C \mid C \in \mathscr{D}\}, init^+(\mathbf{X} \cup \{P_C \mid C \in \mathscr{D}\})),$$

in which a variable P_C for run-time recording the number of parallel copies of each agent $C \in \mathscr{D}$ is available, and prove $\mathscr{A}^+$ is isomorphic to $\mathscr{A}$ (see [9] for further details).

Essentially, the last step is a technical trick that simplifies the overall treatment. The variable P_C counts the number of copies of C present in parallel within the system at a given point in time. To take into account the effects of transitions on agents, we modify updates and rate functions, by increasing/decreasing counter P_C relative to actions adding/removing a copy of C. The reason for introducing state variables will be apparent in next section. They are required to control a cluster of discrete states (continuously approximated) and the *real* value of a state variable will indicate the "tendency" of the system to be in that particular state.

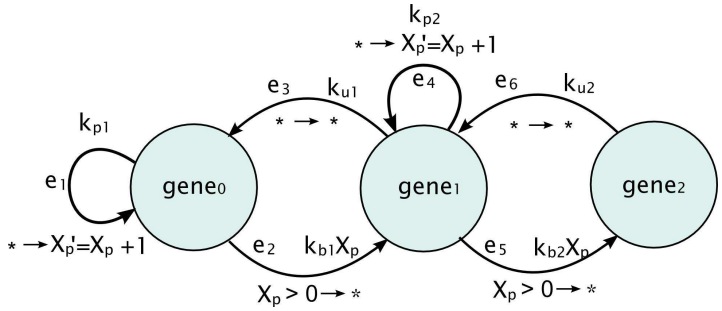


Figure 1: Reduced Transition Systems for the agent $\texttt{gene}_0$ defined in Example 3.1. Each edge is labeled by its rate function and by the guard and the update (depicted in the $guard \rightarrow update$ notation).

4 From sCCP to TDSHA

In this section we define a semantics for **sCCP** in terms of TDSHA. The basic idea is to partition all possible transitions executable by an **sCCP** agent into two classes: those remaining discrete-stochastic and those to be approximated as continuous. Different partitions'schemata correspond to different TDSHA. By parameterizing upon such schemata, we will obtain a *lattice* of different TDSHA's.
Note that this approach is different from [9], as *we do not remove stochasticity*. Indeed, Stochastic Hybrid Automata can be seen as an intermediate layer between stochastic programs and (non-stochastic) hybrid systems. The reader is referred to [8] for further discussions.

The mapping proceeds in two steps. First we convert into TDSHA's each sequential component of a **sCCP** program, then all these TDSHA's are combined using the product construction.

Given a $\mathscr{A}^+ = (A^+, \mathscr{D}^+, \mathbf{Y}, init^+(\mathbf{Y}))$, let $C \in \mathscr{D}^+$ be one of the components of the initial network A, and let $RTS(C) = (S(C), E(C), \ell)$ be its RTS.

A specific continuous/discrete scheme of approximation is formalized by the choice of a boolean vector $\kappa \in \{0,1\}^m$, $m = |E(C)|$, indexed by edges in $E(C)$: for $e \in E(C)$, $\kappa[e] = 1$ stands for a continuous

approximation of the transition, while $\kappa[e] = 0$ implies that the transition will remain discrete. Let $E(\kappa, C) = \{e \in E(C) |\ \kappa[e] = 1\}$ and $E(\neg\kappa, C) = \{e \in E(C) |\ \kappa[e] = 0\}$.

In order to guarantee that the vector field constructed from continuous transitions is sufficiently regular, we identify as *continuously approximable* only those actions π such that $rate(\pi)$ is differentiable and $rate(\pi)[\mathbf{X}] = 0$ whenever $guard(\pi)[\mathbf{X}]$ is false.[6] We call *consistent* a vector κ such that $\kappa[e] = 1$ only for edges e that are continuously approximable. In the following, we suppose to work only with consistent κ.

At this point we are ready to introduce the basic components of our target TDSHA.

Discrete Modes. The modes of the TDSHA will be essentially the states $S(C)$ of the $RTS(C)$. However, as continuous transitions cannot change mode, we need to consider as equivalent those states that can be reached by a path of continuous edges. Let us denote by $\sim_\kappa$ the equivalence relation among states of $S(C)$ relating two states if and only if they are connected by a path of continuous edges (i.e. edges in $E(\kappa, C)$ of the non-oriented version of $RTS(C)$). Let $S_\kappa(C) = S(C)/\sim_\kappa$. For each edge $e \in E(\kappa, C)$, we define the *stoichiometric vector* $\nu_{\mathbf{Y},e}$ as an $|\mathbf{Y}|$-vector, $\mathbf{Y} = \mathbf{X} \cup \{P_C \mid C \in \mathscr{D}\}$, such that $\nu_{\mathbf{Y},e}[X] = h$ if and only if variable X is updated by transition e according to the formula $X' = X + h$.

Example. Consider the gene component of Example 3.1. Its RTS, shown in Figure 1, has three states, corresponding to the three components gene_0, gene_1, and gene_2, with state variables denoted by P_0, P_1, and P_2, respectively. The RTS has also 6 transitions, indexed by $e_1, \ldots, e_6$. Consider the κ vector equal to $(1,0,0,1,1,1)$: edges e_1, e_4, e_5, e_6 will be approximated as continuous, while the other three remain discrete. The relation $\sim_\kappa$ has a quotient state space containing two classes: $S_1 = \{\text{gene}_0\}$ and $S_2 = \{\text{gene}_1, \text{gene}_2\}$. Such a partitioning of the gene's states can be seen as a way to render a slower dynamics for the binding/unbinding mechanism of the first repressor, to be compared to a faster one relative the second copy of the repressor.

Continuous flow. The continuous evolution for TDSHA is given by the following set of continuous transitions:

$$\mathfrak{TC} = \{([exit(e)], \nu_{\mathbf{Y},e}, rate(e) \mid e \in E(\kappa, C)\}.$$

Stochastic transitions. Stochastic transitions are defined in a very simple way, as guards and rates are basically copied from the **sCCP** edge. The only technicality is the definition of the reset.

Consider the state counting variables $\mathbf{P} = \{P_C \mid C \in \mathscr{D}\}$. They can assume values less than or equal to one, as the initial program is *simple*. Moreover, they range in the whole real-valued interval $[0,1]$ whenever we are in a clustered state $[s]$ collapsing $s_1, \ldots, s_k$ of RTS(C). In this case, the state variables $P_{s_1}, \ldots, P_{s_k}$ must sum exactly to 1, their value representing the likelihood of state $s_1, \ldots, s_k$ of the cluster $[s]$, respectively. In order to deal with state clusters correctly, we have to ensure that when a state $[s]$ is left, all its state variables are set to zero. Moreover, if a discrete transition looping in $[s]$ takes place, then the variable of its target state s_i must be set to 1, while all other variables of $[s]$ are to be reset to 0. To enforce this, consider an **sCCP** edge connecting states s_1 and s_2, with

$$update(e) \stackrel{\text{def}}{=} \mathbf{X}' = f(\mathbf{X}) \wedge P'_{s_1} = P_{s_1} - 1 \wedge P'_{s_2} = P_{s_2} + 1,$$

[6] Guards of continuosly approximable π are, in fact, redundant.

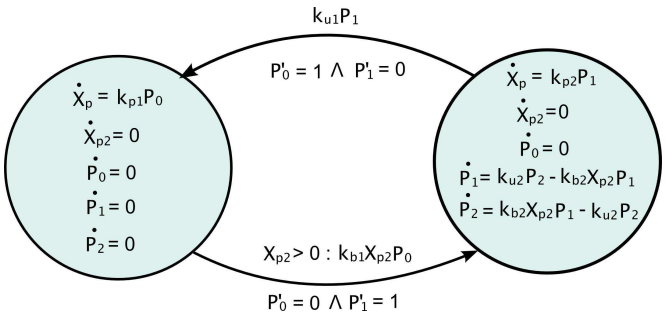


Figure 2: PDMP associated with the gene component of Example 3.1. Transitions approximated continuously determine a set of ODE, while discrete transitions are stochastic and are depicted here as edges of a graph (showing rates, guards and resets). The ODEs are obtained from continuous transitions according to the recipe of Section 2. We chose to display TDSHA in this format as it is similar to the classic representation of hybrid automata.

and define the function f_P on $\mathbf{P}$ which is 1 on the component corresponding to P_{s_2} and zero elsewhere. In this way, $\mathbf{P}' = f_P(\mathbf{P})$ implements the correct updating policy. Let now $\bar{f}$ combine f and f_P: $\bar{f}\begin{pmatrix}\mathbf{X}\\ \mathbf{P}\end{pmatrix} = \begin{pmatrix}f(\mathbf{X})\\ f_P(\mathbf{P})\end{pmatrix}$.

Putting everything together, we have that the discrete transition associated with $e \in E(\neg\kappa, C)$ with $e = (s_1, s_2)$ is

$$([s_1], [s_2], guard(e), \mathbf{Y}' = \bar{f}(\mathbf{Y}), rate(e)) \in \mathfrak{TS}.$$

Instantaneous transitions. At this stage, there is no need to define instantaneous transitions. They will be used in Section 5 to deal with dynamic partitioning.

We can now collect all our considerations into the following definition.

Definition 4.1. Let $\mathscr{A} = (A, \mathscr{D}, \mathbf{X}, init_0)$ be a simple **sCCP** program and $\mathscr{A}^+ = (A^+, \mathscr{D}^+, \mathbf{Y}, init_0^+)$ be its extended version. Let C be a sequential component in parallel in A^+, with $RTS(C) = (S(C), E(C), \ell)$. Fix a boolean vector $\kappa \in \{0,1\}^m$, $m = |E(C)|$. The Transition-Driven Stochastic Hybrid Automaton associated with C with respect to κ is $\mathscr{T}(C, \kappa) = (Q, \mathbf{Y}, \mathfrak{TC}, \mathfrak{TD}, \mathfrak{TS}, init)$, where

- $Q = S_\kappa(C) = S(C)/\sim_\kappa$;
- $\mathfrak{TC} = \{([exit(e)], \nu_{\mathbf{Y},e}, rate(e) \mid e \in E(\kappa, C)\}$;
- $\mathfrak{TD} = \emptyset$;
- $\mathfrak{TS} = \{([s_1], [s_2], guard(e), \mathbf{Y}' = \bar{f}(\mathbf{Y}), rate(e)) \mid e = (s_1, s_2) \in E(\neg\kappa, C)\}$;
- $init = init_0$.

Example. From the previous definition it is easy to generate the TDSHA relative to our running example above, in which $\kappa = (1,0,0,1,1,1)$. Once we have the TDSHA, we can generate the corresponding PDMP (see supplementary material [1]), which is shown in Figure 2.

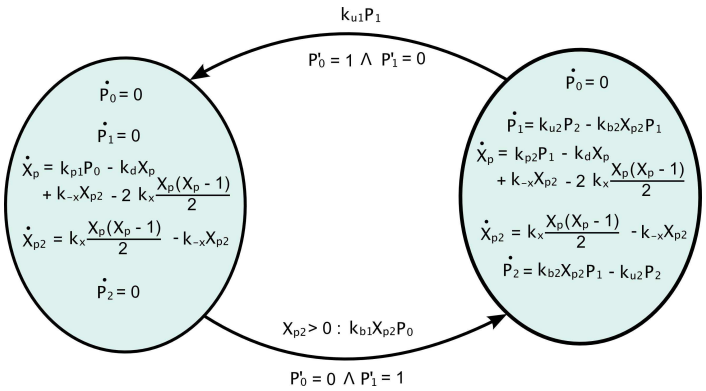


Figure 3: PDMP obtained from the product of TDSHA associated with the three components of the **sCCP** program of Example 3.1. See also the caption of Figure 2.

Definition 4.1 gives a way to associate a TDSHA with a sequential agent of a **sCCP** program. In order to define the TDSHA for the whole program, we will use the product construction.

Definition 4.2. Let $\mathscr{A} = (A, \mathscr{D}, \mathbf{X}, init_0)$ be a simple **sCCP** program and $\mathscr{A}^+ = (A^+, \mathscr{D}^+, \mathbf{Y}, init_0^+)$ be its extended version, with $A^+ = C_1 \parallel \ldots \parallel C_n$. Fix a boolean vector κ_i for each sequential agent C_i. The Transition-Driven Hybrid Automaton for the **sCCP** program $\mathscr{A}$, with respect to $\kappa = (\kappa_i)_{i=1,\ldots n}$ is

$$\mathscr{T}(A, \kappa) = \mathscr{T}(C_1, \kappa_1) \otimes \cdots \otimes \mathscr{T}(C_n, \kappa_n).$$

Example. Consider again the **sCCP** model of Example 3.1. It has three components: gene, deg and dimer, with 6, 1, and 2 edges respectively. We consider three vectors $\kappa_1 = (1,0,0,1,1,2)$, $\kappa_2 = (1)$, and $\kappa_3 = (1,1)$. The product TDSHA of these three components generates the PDMP depicted in Figure 3.

4.1 Lattice of TDSHA

Definition 4.2 associates a TDSHA with a **sCCP** agent for a fixed partition, given by vector κ, of the transitions into discrete and continuous. Clearly, different choices of κ correspond to different TDSHA's, with a different degree of approximation of the original **sCCP** program. The different TDSHA's can be arranged into a lattice according to the following pre-order:

Definition 4.3. Let $\mathscr{A}$ be a **sCCP** agent, then $\mathscr{T}(\mathscr{A}, \kappa_1) \sqsubseteq \mathscr{T}(\mathscr{A}, \kappa_2)$ if and only if $\kappa_1[e] = 1 \Rightarrow \kappa_2[e] = 1$, for each transition $e \in E(A) = E(C_1) \cup \ldots \cup E(C_n)$, with $A = C_1 \parallel \ldots \parallel C_n$ the initial agent of $\mathscr{A}$.

The bottom element of this lattice is obtained for $\kappa \equiv 0$, while the top element is obtained for $\kappa[e] = 1$ if and only if e is continuously approximable. We remind to the reader that transitions not continuously approximable *must* be kept discrete.

The two "extreme" choices correspond to two particularly important TDSHA's, as shown in the following propositions.

Proposition 4.1. *Let $\mathscr{A}$ be a **sCCP** program. The TDSHA $\mathscr{T}(\mathscr{A}, 0)$ is the CTMC associated with $\mathscr{A}$ by its standard semantics.*

Proposition 4.2. *Let $\mathscr{A}$ be a **sCCP** program with initial agent $A = C_1 \parallel \ldots \parallel C_n$. If e is continuously approximable for each $e \in E(A) = E(C_1) \cup \ldots \cup E(C_n)$, $\mathscr{T}(\mathscr{A}, \mathbf{1})$ coincides with the system of ODE's associated with $\mathscr{A}$ by its fluid-flow approximation (see [5]).*

5 Dynamic Partitioning of Transitions

In the previous sections we have defined a mapping from **sCCP** to TDSHA fixing the level of discreteness and continuity. This choice, however, can be difficult to perform *a priori*, as one does not know if the system will evolve to a state where a different approximation is more accurate.
This is particularly true when one deals with biological systems. In this case, reactions involving large populations of molecular species or having high rates, may be treated as continuous. However, such conditions depend on the state of the system and may change during its evolution. Indeed, there has been a growing attention on hybrid simulation strategies in systems biology, cf. next Section.

In order to have dynamic switching, we can extend the discrete modes, introducing states for each admissible vector κ. New discrete transitions need to be added as well, changing the value of κ according to some user defined conditions.
Intuitively, for each transition e the state space must be partitioned in two—possibly empty—regions: one where the edge e is treated as a continuous transition and one in which it is discrete. In order to define such regions, we consider two predicates:

1. $cont[e](\mathbf{X})$, encoding the condition to change edge e from discrete to continuous;
2. $disc[e](\mathbf{X})$, encoding the condition to change edge e from continuous to discrete.

An obvious requirement is that the regions identified by $cont[e]$ and $disc[e]$ should be disjoint. In order to define such predicates, we will consider a (sufficiently regular, usually continuous) function $f_e : \mathbb{R}^n \to \mathbb{R}$, whose sign will discriminate between continuous and discrete regions for edge e. More specifically, we define, for a fixed, small $\varepsilon > 0$.

1. $cont[e](\mathbf{x}) := f_e(\mathbf{x}) \geq \varepsilon$;
2. $disc[e](\mathbf{x}) := f_e(\mathbf{x}) \leq -\varepsilon$.

Using ε instead of 0, not only guarantees that the regions in which $cont[e]$ and $disc[e]$ are true are disjoint, but also avoids pathological situations of infinite sequences of instantaneous transitions. See supplementary material [1] for further details.

Now, suppose e is continuous. If the current trajectory enters in a region of the state space in which $disc[e]$ becomes true, then we must trigger an instantaneous transition in order to move from $\kappa_1[e] = 1$ to $\kappa_2[e] = 0$. All the variables must remain unchanged. However, in this case it may happen that the new relation $\sim_{\kappa_2}$ splits in two the current mode $[s] \in S_{\kappa_1}(C)$, say $[s]_{\kappa_1} = [s_1]_{\kappa_2} \cup [s_2]_{\kappa_2}$. In this case, we need to introduce two instantaneous transitions, one going to $[s_1]$ and the other to $[s_2]$. Now, consider the value of the state variables of $[s]$, $P_{[s]} = \sum_{s_i \in [s]} P_{s_i}$. It can be proved that $P_{[s]} = 1$. Moreover, $P_{[s]} = P_{[s_1]} + P_{[s_2]}$ but, clearly, it is not necessarily the case that the two quantities on the right hand side of the equality are equal. This means that the system may "prefer" to move to states in $[s_1]$ than to those in $[s_2]$. This situation is correctly modeled using priorities, i.e. weighting transition to $[s_i]$ by $P_{[s_i]}$ and re-normalizing variables in $[s_1]$ and $[s_2]$ to maintain the property $P_{[s]} = 1$ for each $[s] \in S_{\kappa_2}(C)$.

We now give a formal definition for this construction, following a similar strategy as in Section 4: first we construct TDSHA for sequential components, then we apply the product construction to combine the local constructions. In order to fix the notation, consider the TDSHA $\mathscr{T}(C, \kappa) = (Q, \mathbf{Y}, \mathfrak{TC}, \mathfrak{TD}, \mathfrak{TS}, init)$ associated with a component C, with respect to a fixed κ. With Q_κ we indicate the set $Q_\kappa = \{([s]_\kappa) \mid [s] \in Q\}$. Moreover, $\mathfrak{TC}_\kappa$, $\mathfrak{TD}_\kappa$, and $\mathfrak{TS}_\kappa$ denote the sets $\mathfrak{TC}$, $\mathfrak{TD}$, and $\mathfrak{TS}$, respectively, with states in Q replaced by the corresponding states (equivalence classes) in Q_κ. A similar rule applies to $init_\kappa$.

Definition 5.1. Let $\mathscr{A} = (A, \mathscr{D}, \mathbf{X}, init_0)$ be a simple **sCCP** program and $\mathscr{A}^+ = (A^+, \mathscr{D}^+, \mathbf{Y}, init_0^+)$ be its extended version. Let C be a sequential agent in parallel in A^+, with $RTS(C) = (S(C), E(C), \ell)$ and

$|E(C)| = m$. Moreover, let $cont[e], disc[e]$, $e \in E(C)$ be defined as above. The TDSHA with dynamic partitioning associated with C is $\mathscr{T}(C, cont, disc) = (Q, \mathbf{Y}, \mathfrak{TC}, \mathfrak{TD}, \mathfrak{TS}, init)$, with:

1. $Q = \bigcup_{\kappa \in \{0,1\}^m} Q_\kappa$;
2. $\mathfrak{TC} = \bigcup_{\kappa \in \{0,1\}^m} \mathfrak{TC}_\kappa$;
3. $\mathfrak{TS} = \bigcup_{\kappa \in \{0,1\}^m} \mathfrak{TS}_\kappa$;
4. $\mathfrak{TD} = \bigcup_{\kappa \in \{0,1\}^m} \mathfrak{TD}_\kappa \cup \mathfrak{TD}_{0,1} \cup \mathfrak{TD}_{1,0}$, where

$$\mathfrak{TD}_{1,0} = \left\{ \left([s_1]_{\kappa_1}, [s_2]_{\kappa_2}, P_{[s_2]}, disc[e], \mathbf{Y}' = g(\mathbf{Y}) \right) \;\middle|\; e \in E(C), \right.$$
$$\left. \kappa_1(e) = 1, \kappa_2(e) = 0, \kappa_1(e') = \kappa_2(e') \; for \; e \neq e', [s_1]_{\kappa_1} \cap [s_2]_{\kappa_2} \neq \emptyset \right\}$$

where g assigns value $\frac{P_{s'}}{P_{[s]}}$ for $s' \in [s_2]_{\kappa_2}$, 0 to any other P_s, and it is the identity on $\mathbf{X}$. Moreover

$$\mathfrak{TD}_{0,1} = \left\{ \left([s]_{\kappa_1}, [s]_{\kappa_2}, 1, cont(e), \mathbf{Y}' = \mathbf{Y} \right) \;\middle|\; e \in E(C), \right.$$
$$\left. \kappa_1(e) = 0, \kappa_2(e) = 1, \kappa_1(e') = \kappa_2(e') \; for \; e \neq e' \right\};$$

5. $init = init_0^+$;

Definition 5.2. Let $\mathscr{A} = (A, \mathscr{D}, \mathbf{X}, init_0)$ be a simple **sCCP** program and $\mathscr{A}^+ = (A^+, \mathscr{D}^+, \mathbf{Y}, init_0^+)$ be its extended version, with $A^+ = C_1 \parallel \ldots \parallel C_n$. Moreover, fix predicates $cont_j[e], disc_j[e]$ for each sequential agent C_j of A^+, according to Definition 5.1. The Transition-Driven Stochastic Hybrid Automata with dynamic partitioning for the **sCCP** program $\mathscr{A}$, with respect to $(cont_j, disc_j)_{j=1,\ldots n}$ is

$$\mathscr{T}(A, (cont_j, disc_j)_{j=1,\ldots n}) = \mathscr{T}(C_1, cont_1, disc_1) \otimes \cdots \otimes \mathscr{T}(C_n, cont_n, disc_n).$$

Remark 5.1 (On the fly simulation). In Definition 5.2, the resulting TDHA has a number of modes exponential in the number of transitions that sequential agents can perform. This combinatorial explosion rules out the possibility of generating all the modes together. However, if we restrict to simulation, this is not a real issue, as we need to record only the current mode: the target mode of a transition can be generated on the fly as soon as the transition has been taken, given the knowledge of RTS.

5.1 Hybrid Simulation Strategies

The hybrid simulation algorithms proposed in literature [19, 24, 20, 28, 23, 16, 2] basically differ in two aspects: the kind of continuous dynamics (it can be based on ODE or SDE) and the rules for partitioning reactions into continuous and discrete (usually called fast and slow). More specifically, the partitioning can be static (done at the beginning of the simulation) or dynamic (i.e. recomputed at run-time).

Conditions for separating fast and slow reactions are usually twofold:

1. the *size of species* involved in the reaction must all be bigger than a given threshold. Usually, a fast reaction j must satisfy a condition like $x_i \geq K|\nu_{i,j}|$ for all species i involved in j, where ν is the stoichiometric matrix.

2. the *rate function* of fast reactions must be reasonably bigger than that of slow reactions. Usually, the following constraint is enforced [28]: $\lambda_j(\mathbf{x})\Delta t \geq \Lambda$, which ensures that reaction j fires many times during the time step Δt. In [16] a different partition strategy imposes that rates of fast reactions are Λ times faster than the fastest slow reaction, so as to guarantee a separation of time scales.

Dynamical policies sketched above can be easily accounted for in our setting.
First of all, we need to start from an **sCCP** model of a biochemical network [6], in which reactions are modeled by action capabilities of agents. Then, applying the framework of this paper, we associate a TDSHA with such a model, together with a suitable policy for dynamic partitioning of transitions. All we have to do is define a function f_e for each **sCCP** transition e, such that $f_e(\mathbf{x}) > 0$ when the associated reaction can be considered fast and $f_e(\mathbf{x}) < 0$ when it is slow.

As an example, consider a partition strategy based only on the size of populations, like the one adopted in [24]. In this case, the function f_e for transition e can be the following:

$$f_e(\mathbf{x}) = \min\{x_i - K|\nu[x_i, e]| \mid \nu[x_i, e] \neq 0\},$$

where $\nu[\cdot, e]$ is the stoichiometry of action e, constructed as in Section 4, and K is a constant (that can be tuned for the specific system). Of course, more complex policies can be introduced by suitably modifying the functions f_e.

6 Conclusion and Further Directions

In this paper we provided a specific process algebra, **sCCP**, with a general semantics based on stochastic hybrid systems, parametric with respect to the degree of continuity and discreteness. The different hybrid models generated in this way can be arranged in a lattice, and we provided also a way to dynamically move within the lattice. This allows to formally describe hybrid simulation algorithms, opening up their use as tools to simulate process algebra-based models. Moreover, this approach gives the possibility of using other computational analysis methods than simulation, like reachability computations or model checking.
An interesting problem is how to extend such machinery to other process algebras. First steps have been done to deal with stochastic π-calculus [7], however the peculiarities of each language present specific difficulties to be solved.

The formal treatment developed in the paper, in particular the lattice of TDSHA defined in Section 4.1, can also provide an interesting theoretical framework to study the quality of the approximation and the error introduced. In particular, the mature theory of PDMP [12] can provide interesting tools in this direction.

Another issue we are investigating regards the relationships between discreteness and stochasticity. In particular, we are interested in understanding whether the stochastic ingredient of the dynamics can be dropped in favor of a pure discrete evolution, and at what price [8]. Motivations for this reside in the fact that non-stochastic hybrid systems have a much wider and more efficient set of automated reasoning tools available.

We conclude with a more basic (perhaps philosophical) question: given that a mix of continuous and discrete simulation strategy is the choice, is there a way—other than minimization of computational complexity—to determine which parts of the systems can/may be simulated discretely/continuously? We feel that physical consideration must be taken into account for addressing this issue and that these are probably outside our reach. However we wish to contribute the "computer scientist point of view":

the level of discreteness/continuity can be established on the ground of a formal specification of the properties to verify/simulate and should guarantee the minimum of computational resources necessary to this task.

References

[1] Supplementary matherial to the paper available online at: `http://www.dmi.units.it/~bortolu/sccp.htm`.

[2] A. Alfonsi, E. Cances, G. Turinici, B. Di Ventura & W. Huisinga (2005): *Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems*. In: *Proceedings of ESAIM*, 14. pp. 1–13.

[3] L. Bortolussi (2006): *Stochastic Concurrent Constraint Programming*. In: *Proceedings of 4th International Workshop on Quantitative Aspects of Programming Languages (QAPL 2006)*, *ENTCS* 164. pp. 65–80.

[4] L. Bortolussi & A. Policriti (2009): *Dynamical systems and stochastic programming - from Ordinary Differential Equations and back*. *Transactions of Computational Systems Biology, in print.*

[5] L. Bortolussi & A. Policriti (2007): *Stochastic Concurrent Constraint Programming and Differential Equations*. In: *Proceedings of Fifth Workshop on Quantitative Aspects of Programming Languages, QAPL 2007*, *ENTCS* 167.

[6] L. Bortolussi & A. Policriti (2008): *Modeling Biological Systems in Concurrent Constraint Programming*. *Constraints* 13(1).

[7] L. Bortolussi & A. Policriti (2009): *Hybrid Dynamics of Stochastic π-calculus*. *Mathematics in Computer Science* 2(3), pp. 465–491.

[8] L. Bortolussi & A. Policriti (2009): *Hybrid Dynamics of Stochastic Programs*. *Submitted to Theor. Comp. Sc.* .

[9] L. Bortolussi & A. Policriti (2009): *Stochastic Programs and Hybrid Automata for (Biological) Modeling*. In: *Proceedings of CiE 2009.*

[10] M.L. Bujorianu & J. Lygeros (2004): *General Stochastic Hybrid Systems: Modeling and Optimal Control*. In: *Proceedings of 43rd IEEE Conference on Decision and Control (CDC 2004)*. pp. 182–187.

[11] M. Calder, S. Gilmore & J. Hillston (2006): *Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA*. *Transactions on Computational Systems Biology* 4230, pp. 1–23.

[12] M.H.A. Davis (1993): *Markov Models and Optimization*. Chapman & Hall.

[13] D. Gillespie (2000): *The chemical Langevin equation*. *Journal of Chemical Physics* 113(1), pp. 297–306.

[14] D.T. Gillespie (1976): *A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions*. *J. of Computational Physics* 22.

[15] D.T. Gillespie (1977): *Exact Stochastic Simulation of Coupled Chemical Reactions*. *J. of Physical Chemistry* 81(25).

[16] M. Griffith, T. Courtney, J. Peccoud & W.H. Sanders (2006): *Dynamic partitioning for hybrid simulation of the bistable HIV-1 transactivation network*. *Bioinformatics* 22(22), pp. 2782–2789.

[17] V. Gupta, R. Jagadeesan & P. Panangaden (1999): *Stochastic processes as concurrent constraint programs*. In: *Proceedings of POPL'99.*

[18] V. Gupta, R. Jagadeesan & V.A. Saraswat (1997): *Probabilistic Concurrent Constraint Programming*. In: *Proceedings of CONCUR'97.*

[19] E.L. Haseltine & J.B. Rawlings (2002): *Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics*. *Journal of Chemical Physics* 117(15).

[20] E.L. Haseltine & J.B. Rawlings (2005): *On the origins of approximations for stochastic chemical kinetics*. *J. Chem. Phys.* 123.

[21] T. A. Henzinger (1996): *The theory of hybrid automata*. In: *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science.*

[22] J. Hillston (2005): *Fluid Flow Approximation of PEPA models*. In: *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST05)*.

[23] T.R. Kiehl, R.M. Mattheyses & M.K. Simmons (2004): *Hybrid Simulation of Cellular Behavior*. *Bioinformatics* 20(3), pp. 316–322.

[24] N. A. Neogi (2004): *Dynamic Partitioning of Large Discrete Event Biological Systems for Hybrid Simulation and Analysis*. In: *Proceedings of 7th International Workshop on Hybrid Systems: Computation and Control, HSCC 2004, LNCS* 2993. pp. 463–476.

[25] J. Pahle (2009): *Biochemical simulations: stochastic, approximate stochastic and hybrid approaches*. *Brief Bioinform.* 10(1), pp. 53–64.

[26] A. Di Pierro & H. Wiklicky (1998): *An operational semantics for probabilistic concurrent constraint programming*. In: *Proceedings of IEEE Computer Society International Conference on Computer Languages*.

[27] A. Regev & E. Shapiro (2002): *Cellular Abstractions: Cells as Computation*. *Nature* 419.

[28] H. Salis & Y. Kaznessis (2005): *Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions*. *Journal of Chemical Physics* 122.

[29] V. Saraswat & M. Rinard (1990): *Concurrent Constraint Programming*. In: *Proceedings of 18th Symposium on Principles Of Programming Languages (POPL)*.

[30] V. A. Saraswat (1993): *Concurrent Constraint Programming*. MIT press.

[31] S. Strubbe & A. van der Schaft (2007): *Stochastic Hybrid Systems*, chapter Compositional Modeling of Stochastic Hybrid Systems, pp. 47–78. CRC Press.

[32] D. J. Wilkinson (2006): *Stochastic Modelling for Systems Biology*. Chapman & Hall.

Modelling an Ammonium Transporter with SCLS*

Mario Coppo[1], Ferruccio Damiani[1], Elena Grassi[1,2], Mike Guether[3] and Angelo Troina[1]

[1]Dipartimento di Informatica, Università di Torino

[2]Molecular Biotechnology Center, Dipartimento di Genetica, Biologia e Biochimica, Università di Torino

[3]Dipartimento di Biologia Vegetale, Università di Torino

{coppo,damiani,troina}@di.unito.it grassi.e@gmail.com mike.guether@unito.it

The Stochastic Calculus of Looping Sequences (SCLS) is a recently proposed modelling language for the representation and simulation of biological systems behaviour. It has been designed with the aim of combining the simplicity of notation of rewrite systems with the advantage of compositionality. It also allows a rather simple and accurate description of biological membranes and their interactions with the environment.

In this work we apply SCLS to model a newly discovered ammonium transporter. This transporter is believed to play a fundamental role for plant mineral acquisition, which takes place in the arbuscular mycorrhiza, the most wide-spread plant-fungus symbiosis on earth. Due to its potential application in agriculture this kind of symbiosis is one of the main focuses of the BioBITs project.

In our experiments the passage of NH_3 / NH_4^+ from the fungus to the plant has been dissected in known and hypothetical mechanisms; with the model so far we have been able to simulate the behaviour of the system under different conditions. Our simulations confirmed some of the latest experimental results about the LjAMT2;2 transporter. The initial simulation results of the modelling of the symbiosis process are promising and indicate new directions for biological investigations.

1 Introduction

Given the central role of agriculture in worldwide economy, several ways to optimize the use of costly artificial fertilizers are now being actively pursued. One approach is to find methods to nurture plants in more "natural" manners, avoiding the complex chemical production processes used today. In the last decade the Arbuscular Mycorrhiza (AM), the most widespread symbiosis between plants and fungi, got into the focus of research because of its potential as a natural plant fertilizer. Briefly, fungi help plants to acquire nutrients as phosphorus (P) and nitrogen (N) from the soil whereas the plant supplies the fungus with energy in form of carbohydrates [32]. The exchange of these nutrients is supposed to occur mainly at the eponymous arbuscules, a specialized fungal structure formed inside the cells of the plant root. The arbuscules are characterized by a juxtaposition of a fungal and a plant cell membrane where a very active interchange of nutrients is facilitated by several membrane transporters. These transporters are surface proteins that facilitate membrane crossing of molecules which, because of their inherent chemical nature, are not freely diffusible.

As long as almost each cell in the majority of multicellular organisms shares the same genome, modern theories point out that morphological and functional differences between them are mainly driven by different genes expression [2]. Thanks to the last experimental novelties [40, 30] a precise analysis of which genes are expressed in a single tissue is possible; therefore is possible to identify genes that are pivotal in specific compartments and then study their biological function. Following this route a new

*This research is founded by the BioBITs Project (*Converging Technologies* 2007, area: Biotechnology-ICT), Regione Piemonte.

R.J.Back, I.Petre, E. de Vink (Eds.): Computational Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 77–92, doi:10.4204/EPTCS.6.6

membrane transporter has been discovered by expression analysis and further characterized [18]. This transporter is situated on the plant cell membrane which is directly opposed to the fungal membrane, located in the arbuscules. Various experimental evidence points out that this transporter binds to an NH_4^+ moiety outside the plant cell, deprotonates it, and mediates inner transfer of NH_3, which is then used as a nitrogen source, leaving an H^+ ion outside. The AM symbiosis is far from being unraveled: the majority of fungal transporters and many of the chemical gradients and energetic drives of the symbiotic interchanges are unknown. Therefore, a valuable task would be to model *in silico* these conditions and run simulations against the experimental evidence available so far about this transporter. Conceivably, this approach will provide biologists with working hypotheses and conceptual frameworks for future biological validation.

In computer science, several formalisms have been proposed for the description of the behaviour of biological systems. Automata-based models [3, 28] have the advantage of allowing the direct use of many verification tools such as model checkers. Rewrite systems [11, 35, 8, 1] usually allow describing biological systems with a notation that can be easily understood by biologists. Compositionality allows studying the behaviour of a system componentwise. Both automata-like models and rewrite systems present, in general, problems from the point of view of compositionality, which, instead, is in general ensured by process calculi, included those commonly used to describe biological systems [36, 34, 9].

The Stochastic Calculus of Looping Sequences (SCLS) [6] (see also [8, 7, 29]) is a recently proposed modelling language for the representation and simulation of biological systems behaviour. It has been designed with the aim of combining the simplicity of notation of rewrite systems with the advantage of a form of compositionality. It also allows a rather simple and accurate description of biological membranes and their interactions with the environment.

In this work we apply SCLS to model the mentioned transporter. This transporter is differentially expressed in arbuscular cells and is believed to play a fundamental role in the nutrients uptake which takes place in the context of plants-fungi symbiosis at the root level. This symbiosis is one of the main focuses of the BioBITs project, due to its relevant role in agriculture.

On these premises, the aim of this work is to model the interchange between the fungus-plant interface and the plant cells, using as a reference system the NH_3/NH_4^+ turnover. This could disclose which is the driving power of the net nitrogen flux inside plant cells (still unknown at the chemical level). Furthermore, this information may also be exploited to model other, and so far poorly characterized, transporters.

Outline Section 2 introduces the syntax and the semantics of the SCLS and shows some modelling guidelines. Section 3 presents the SCLS representations of the ammonium transporter and our experimental results, discussed in Section 4. Section 5 concludes by outlining some further work.

2 The Stochastic Calculus of Looping Sequences

In this section we briefly recall the Stochastic Calculus of Looping Sequences (SCLS) [6]. A SCLS (biological) model consists of a term, representing the structure of the modelled system and a set of stochastic rewrite rules which determine its evolution.

Terms Terms of SCLS are defined from a set of atomic symbols representing the basic elements of the system under description, which could correspond, according to the level of the description, to genes, proteins, molecules or even single chemical elements. Terms are built from the atomic elements via

the operators of sequencing (defining an ordered sequence of atomic elements), looping (defining membranes from looping sequences) and parallel composition (representing coexistence in the same biological ambient).

Assuming a possibly infinite alphabet $\mathscr{E}$ of symbols ranged over by $a, b, c, \ldots$ the syntax of *SCLS terms* T and *sequences* S is defined by the grammar:

$$\begin{array}{rcl} T & ::= & S \mid (S)^L \rfloor T \mid T \mid T \\ S & ::= & \varepsilon \mid a \mid S{\cdot}S \end{array}$$

where a is any element of $\mathscr{E}$ and ε is the empty sequence. We denote the infinite sets of terms and sequences with $\mathscr{T}$ and $\mathscr{S}$, respectively.

The SCLS terms are built from the elements of $\mathscr{E}$ by means of four syntactic operators: sequencing $\_\cdot\_$, looping $(\_)^L$, parallel composition $\_\,|\,\_$, and the containment operator $\_\rfloor\_$. Sequencing is used to concatenate atomic elements. The looping operator transforms a sequence in a closed loop when combined with another term via the $\_\rfloor\_$ operator. A term can then be either a sequence, or a looping sequence containing another term, or the parallel composition of two terms. By the definition of terms, we have that looping and containment are always associated, hence we can consider $(\_)^L\rfloor\_$ as a single binary operator that applies to one sequence and one term.

The biological interpretation of the operators is the following: the main entities which occur in cells are DNA and RNA strands, proteins, membranes, and other macro-molecules. DNA strands (and similarly RNA strands) are sequences of nucleic acids, but they can be seen also at a higher level of abstraction as sequences of genes. Proteins are sequences of amino acids which typically have a very complex three-dimensional structure. In a protein there are usually (relatively) few subsequences, called domains, which actually are able to interact with other entities by means of chemical reactions. SCLS sequences can model DNA/RNA strands and proteins by describing each gene or each domain with a symbol of the alphabet. Membranes are closed surfaces often interspersed with proteins, and may have a content. A closed surface can be modelled by a looping sequence. The elements (or the subsequences) of the looping sequence may represent the proteins on the membrane, and by the containment operator it is possible to specify what the membrane contains. Other macro-molecules can be modeled as single alphabet symbols, or as sequences of their components. Finally, juxtaposition of entities can be described by the parallel composition operator of their representations. More detailed description of the biological interpretation of the SCLS operators involved in the description of the system presented in this paper will be given at the end of this section.

Brackets can be used to indicate the order of application of the operators. We assume that $(\_)^L\rfloor\_$ has the precedence over $\_\,|\,\_$ and $\_\cdot\_$ over $\_\,|\,\_$, therefore $(S)^L\rfloor T_1\,|\,T$ has to be read as $((S)^L\rfloor T_1)\,|\,T$ and $a\cdot b\,|\,c$ as $(a\cdot b)\,|\,c$. An example of an SCLS term is $a\,|\,b\,|\,(m\cdot n)^L\rfloor(c\cdot d\,|\,e)$ consisting of three entities a, b and $(m\cdot n)^L\rfloor(c\cdot d\,|\,e)$. It represents a membrane with two molecules m and n (for instance, two proteins) on its surface, and containing a sequence $c\cdot d$ and a molecule e. Molecules a and b are outside the membrane. See Figure 1 for some graphical representations.

Structural congruence As usual in systems of this kind, SCLS terms representing the same entity may have syntactically different structures. Such terms are thus identified by means of a *structural congruence* relation. In particular the structural congruence relations $\equiv_S$ and $\equiv_T$ are defined as the least

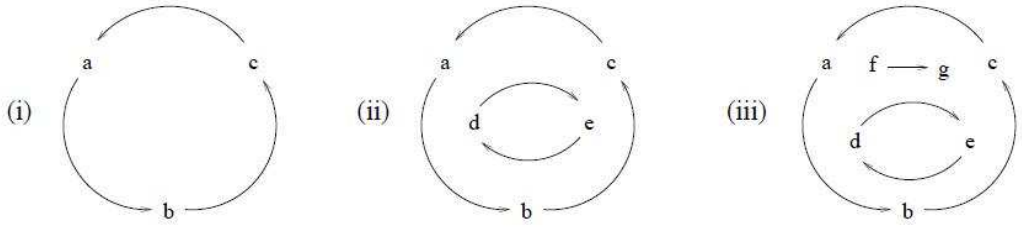


Figure 1: (i) $(a{\cdot}b{\cdot}c)^L \rfloor \varepsilon$; (ii) $(a{\cdot}b{\cdot}c)^L \rfloor (d{\cdot}e)^L \rfloor \varepsilon$; (iii) $(a{\cdot}b{\cdot}c)^L \rfloor (((d{\cdot}e)^L \rfloor \varepsilon) \,|\, f{\cdot}g)$.

congruence relations on sequences and on terms, respectively, satisfying the following rules:

$$S_1{\cdot}(S_2{\cdot}S_3) \equiv_S (S_1{\cdot}S_2){\cdot}S_3 \qquad S{\cdot}\varepsilon \equiv_S \varepsilon{\cdot}S \equiv_S S$$
$$S_1 \equiv_S S_2 \text{ implies } S_1 \equiv_T S_2 \text{ and } (S_1)^L \rfloor T \equiv_T (S_2)^L \rfloor T$$
$$T_1 \,|\, T_2 \equiv_T T_2 \,|\, T_1 \quad T_1 \,|\, (T_2 \,|\, T_3) \equiv_T (T_1 \,|\, T_2) \,|\, T_3 \quad T \,|\, \varepsilon \equiv_T T$$
$$T_1 \equiv_T T_2 \text{ implies } (S)^L \rfloor T_1 \equiv_T (S)^L \rfloor T_2$$
$$(\varepsilon)^L \rfloor \varepsilon \equiv_T \varepsilon \qquad (S_1{\cdot}S_2)^L \rfloor T \equiv_T (S_2{\cdot}S_1)^L \rfloor T$$

Structural congruence states the associativity of $\cdot$ and $|$, the commutativity of the latter and the neutral role of ε. Moreover, axiom $(S_1{\cdot}S_2)^L \rfloor T \equiv_T (S_2{\cdot}S_1)^L \rfloor T$ says that looping sequences can rotate. In the following we will simply use $\equiv$ in place of $\equiv_T$, $\equiv_S$.

Rewrite Rules, Variables and Patterns A rewrite rule is defined as a pair of terms (possibly containing variables), which represent the patterns defining the system transformations, together with a rate representing the speed of the modelled reaction. Rules are applicable to all subterms, identified by the notion of reduction context introduced below, which match the left-hand side of the rule via a proper instantiation of its variables. The system transformation is obtained by replacing the reduced subterm by the corresponding instance of the right-hand side of the rule.

Variables in patterns can be of three kinds: two are associated with the two different syntactic categories of terms and sequences, and one is associated with single alphabet elements. We assume a set of term variables TV ranged over by $X, Y, Z, \ldots$, a set of sequence variables SV ranged over by $\widetilde{x}, \widetilde{y}, \widetilde{z}, \ldots$, and a set of element variables $\mathscr{X}$ ranged over by $x, y, z, \ldots$. All these sets are pairwise disjoint and possibly infinite. We denote by $\mathscr{V}$ the set of all variables $TV \cup SV \cup \mathscr{X}$, and with ρ any variable in $\mathscr{V}$. A pattern is a term which may include variables, i.e. *patterns* P and *sequence patterns* SP of SCLS are given by the following grammar:

$$\begin{array}{rcl} P & ::= & SP \;\mid\; (SP)^L \rfloor P \;\mid\; P \,|\, P \;\mid\; X \\ SP & ::= & \varepsilon \;\mid\; a \;\mid\; SP{\cdot}SP \;\mid\; \widetilde{x} \;\mid\; x \end{array}$$

where a is an element of $\mathscr{E}$, and $X, \widetilde{x}$ and x are elements of TV, SV and $\mathscr{X}$, respectively. The infinite set of patterns is denoted with $\mathscr{P}$. The structural congruence relation can be be trivially extended to patterns. An *instantiation* is a partial function $\sigma : \mathscr{V} \to \mathscr{T}$ which preserves the type of variables, thus for $X \in TV, \widetilde{x} \in SV$ and $x \in \mathscr{X}$ we must have $\sigma(X) \in \mathscr{T}$, $\sigma(\widetilde{x}) \in \mathscr{S}$, and $\sigma(x) \in \mathscr{E}$, respectively. Given $P \in \mathscr{P}$, the expression $P\sigma$ denotes the term obtained by replacing each occurrence of each variable $\rho \in \mathscr{V}$ appearing in P with the corresponding term $\sigma(\rho)$.

Let Σ denote the set of all the possible instantiations and $Var(P)$ the set of variables appearing in $P \in \mathscr{P}$. Then a *rewrite rule* is a triple (P_1, P_2, k), denoted with $P_1 \overset{k}{\mapsto} P_2$, where $k \in \mathbb{R}^{\geq 0}$ is the kinetic constant of the modeled chemical reaction, $P_1, P_2 \in \mathscr{P}$, $P_1 \not\equiv \varepsilon$ such that $Var(P_2) \subseteq Var(P_1)$. A rewrite rule $P_1 \overset{k}{\mapsto} P_2$ then states that a subterm $P_1\sigma$, obtained by instantiating variables in P_1 by some instantiation function σ, can be transformed into the subterm $P_2\sigma$ with kinetics given by the constant k.

Contexts The definition of reduction for SCLS systems is completed by resorting to the notion of reduction context. To this aim, as usual, the syntax of terms is enriched with a new element $\square$ representing a hole. *Reduction context* (ranged over by C) are defined by:

$$C \ ::= \ \square \quad | \quad C \,|\, T \quad | \quad T \,|\, C \quad | \quad (S)^L \rfloor C$$

where $T \in \mathscr{T}$ and $S \in \mathscr{S}$. The context $\square$ is called the *empty context*. We denote with $\mathscr{C}$ the infinite set of contexts.

By definition, every context contains a single $\square$. Let us assume $C, C' \in \mathscr{C}$. Then $C[T]$ denotes the term obtained by replacing $\square$ with T in C; similarly $C[C']$ denotes context composition, whose result is the context obtained by replacing $\square$ with C' in C. The structural equivalence is extended to contexts in the natural way (i.e. by considering $\square$ as a new and unique symbol of the alphabet $\mathscr{E}$).

Note that the general form of rewrite rules does not permit to have sequences as contexts. A rewrite rule introducing a parallel composition on the right hand side (as $a \overset{k}{\mapsto} b \,|\, c$) applied to an element of a sequence (e.g., $m \cdot a \cdot m$) would result into a syntactically incorrect term (in this case $m \cdot (b \,|\, c) \cdot m$); which moreover does not seem to have any biological meaning, parallel and sequence represent orthogonal notions. Therefore to modify a sequence, a pattern representing the whole sequence must appear in the rule. For example, rule $a \cdot \widetilde{x} \overset{k}{\mapsto} a \,|\, \widetilde{x}$ can be applied to any sequence starting with element a, and, hence, the term $a \cdot b$ can be rewritten as $a \,|\, b$, and the term $a \cdot b \cdot c$ can be rewritten as $a \,|\, b \cdot c$. The notion of reduction context will forbid that a substitution like this is applied to looping sequences.

Stochastic Reduction Semantics The operational semantics of SCLS is defined by incorporating a collision-based stochastic framework along the line of the one presented by Gillespie in [13], which is, *de facto*, the standard way to model quantitative aspects of biological systems. Following the law of mass action, it is necessary to count the number of reactants that are present in a system in order to compute the exact rate of a reaction. The same approach has been applied, for instance, to the stochastic π-calculus [33, 34]. The idea of Gillespie's algorithm is that a rate constant is associated with each considered chemical reaction. Such a constant is obtained by multiplying the kinetic constant of the reaction by the number of possible combinations of reactants that may occur in the system. The resulting rate is then used as the parameter of an exponential distribution modelling the time spent between two occurrences of the considered chemical reaction.

The use of exponential distributions to represent the (stochastic) time spent between two occurrences of chemical reactions allows describing the system as a Continuous Time Markov Chain (CTMC), and consequently allows verifying properties of the described system analytically and by means of stochastic model checkers.

The number of reactants in a reaction represented by a reduction rule is evaluated considering the number of occurrences of subterms to which the rule can be applied and of the terms produced. For instance in evaluating the application rate of the rewrite rule $R = a \,|\, b \overset{k}{\mapsto} c$ to the term $T = a \,|\, a \,|\, b \,|\, b$ we

must consider the number of the possible combinations of reactants of the form $a \mid b$ in T. Since each occurrence of a can react with each occurrence of b, this number is 4. So the application rate of R is $k \cdot 4$.

The evaluation of the application rate of a reduction rule containing variables is more complicate since there can be many different ways in which variables can be instantiated to match the subterm to be reduced, and this must be considered to correctly evaluate the application rate. The technique to do this is described in [6]. With this technique, given two terms T, T' and a reduction rule R, we can compute the function $\mathscr{O}(R,T,T')$, defining the number of the possible applications of the rule R to the term T resulting in the term T'. We refer to [6] for more details and explanation.

Given a finite set $\mathscr{R}$ of stochastic rewrite rules, then, the *reduction semantics* of SCLS is the least labelled transition relation satisfying the following rule:

$$\frac{R = P_1 \overset{k}{\mapsto} P_2 \in \mathscr{R} \quad T \equiv C[P_1\sigma] \quad T' \equiv C[P_2\sigma]}{T \xrightarrow{k \cdot \mathscr{O}(R,T,T')} T'}$$

The rate of the reduction is then obtained as the product of the rewrite rate constant and the number of occurrences of the rule within the starting term (thus counting the exact number of reactants to which the rule can be applied and which produce the same result). The rate associated with each transition in the stochastic reduction semantics is the parameter of an exponential distribution that characterizes the stochastic behaviour of the activity corresponding to the applied rewrite rule. The stochastic semantics is essentially a *Continuous Time Markov Chain* (CTMC). A standard simulation procedure that corresponds to Gillespie's simulation algorithm [13] can be followed. The most recent implementation of SCLS, based on Gillespie's algorithm, is described in [38].

Modelling Guidelines SCLS can be used to model biomolecular systems analogously to what is done, e.g, by Regev and Shapiro in [37] for the π-calculus. An abstraction is a mapping from a real-world domain to a mathematical domain, which may allow highlighting some essential properties of a system while ignoring other, complicating, ones. In [37], Regev and Shapiro show how to abstract biomolecular systems as concurrent computations by identifying the biomolecular entities and events of interest and by associating them with concepts of concurrent computations such as concurrent processes and communications.

The use of rewrite systems, such as SCLS, to describe biological systems is founded on a different abstraction. Usually, entities (and their structures) are abstracted by terms of the rewrite system, and events by rewrite rules. We have already recalled the biological interpretation of SCLS operators in the previous section.

In order to describe cells, it is quite natural to consider molecular populations and membranes. Molecular populations are groups of molecules that are in the same compartment of the cell. As we have said before, molecules can be of many types: they could be classified as DNA and RNA strands, proteins, and other molecules. Membranes are considered as elementary objects, in the sense that we do not describe them at the level of the lipids they are made of. The only interesting properties of a membrane are that it may have a content (hence, create a compartment) and that it may have molecules on its surface.

We give now some examples of biomolecular events of interest and their description in SCLS. The simplest kind of event is the change of state of an elementary object. Then, we consider interactions between molecules: in particular complexation, decomplexation and catalysis. These interactions may involve single elements of non-elementary molecules (DNA and RNA strands, and proteins). Moreover,

Biomolecular Event	Examples of SCLS Rewrite Rule
State change	$a \mapsto b$ $\widetilde{x}\cdot a\cdot\widetilde{y} \mapsto \widetilde{x}\cdot b\cdot\widetilde{y}$
Complexation	$a\,|\,b \mapsto c$ $\widetilde{x}\cdot a\cdot\widetilde{y}\,|\,b \mapsto \widetilde{x}\cdot c\cdot\widetilde{y}$
Decomplexation	$c \mapsto a\,|\,b$ $\widetilde{x}\cdot c\cdot\widetilde{y} \mapsto \widetilde{x}\cdot a\cdot\widetilde{y}\,|\,b$
Catalysis	$c\,|\,P_1 \mapsto c\,|\,P_2$ (where $P_1 \mapsto P_2$ is the catalyzed event)
Membrane crossing	$a\,|\,(\widetilde{x})^L\rfloor X \mapsto (\widetilde{x})^L\rfloor(a\,|\,X)$ $(\widetilde{x})^L\rfloor(a\,|\,X) \mapsto a\,|\,(\widetilde{x})^L\rfloor X$ $\widetilde{x}\cdot a\cdot\widetilde{y}\,|\,(\widetilde{z})^L\rfloor X \mapsto (\widetilde{z})^L\rfloor(\widetilde{x}\cdot a\cdot\widetilde{y}\,|\,X)$ $(\widetilde{z})^L\rfloor(\widetilde{x}\cdot a\cdot\widetilde{y}\,|\,X) \mapsto \widetilde{x}\cdot a\cdot\widetilde{y}\,|\,(\widetilde{z})^L\rfloor X$
Catalyzed membrane crossing	$a\,|\,(b\cdot\widetilde{x})^L\rfloor X \mapsto (b\cdot\widetilde{x})^L\rfloor(a\,|\,X)$ $(b\cdot\widetilde{x})^L\rfloor(a\,|\,X) \mapsto a\,|\,(b\cdot\widetilde{x})^L\rfloor X$ $\widetilde{x}\cdot a\cdot\widetilde{y}\,|\,(b\cdot\widetilde{z})^L\rfloor X \mapsto (b\cdot\widetilde{z})^L\rfloor(\widetilde{x}\cdot a\cdot\widetilde{y}\,|\,X)$ $(b\cdot\widetilde{z})^L\rfloor(\widetilde{x}\cdot a\cdot\widetilde{y}\,|\,X) \mapsto \widetilde{x}\cdot a\cdot\widetilde{y}\,|\,(b\cdot\widetilde{z})^L\rfloor X$
Membrane joining	$(\widetilde{x})^L\rfloor(a\,|\,X) \mapsto (a\cdot\widetilde{x})^L\rfloor X$ $(\widetilde{x})^L\rfloor(\widetilde{y}\cdot a\cdot\widetilde{z}\,|\,X) \mapsto (\widetilde{y}\cdot a\cdot\widetilde{z}\cdot\widetilde{x})^L\rfloor X$
Catalyzed	$(b\cdot\widetilde{x})^L\rfloor(a\,|\,X) \mapsto (a\cdot b\cdot\widetilde{x})^L\rfloor X$

Table 1: Guidelines for the abstraction of biomolecular events into SCLS.

there can be interactions between membranes and molecules: in particular a molecule may cross or join a membrane.

Table 1 lists the guidelines (taken from [6]) for the abstraction into SCLS rules of the biomolecular events we will use in our application.[1] Entities are associated with SCLS terms: elementary objects are modelled as alphabet symbols, non-elementary objects as SCLS sequences and membranes as looping sequences. Biomolecular events are associated with SCLS rewrite rules.

3 Modelling the Ammonium Transporter with SCLS

The scheme in Figure 2 (taken from [18]) illustrates nitrogen, phosphorus and carbohydrate exchanges at the mycorrhizal interface according to previous works and the results of [18]. In this paper we focus our investigation on the sectors labelled with **(c)**, **(a1)** and **(a2)**. Namely, we will present SCLS models for the equilibrium between NH_4^+ and NH_3 and the uptake by the LjAMT2;2 transporter **(c)**, and the exchange of NH_4^+ from the fungus to the interspatial level **(a1-2)**. The choice of SCLS is motivated by the fact that membranes, membrane elements (like LjAMT2;2) and the involved reactions can be represented in it in a quite natural way.

The simulations illustrated in this section are done with the SCLSm prototype simulator [38]. In the following we will use a more compact notation for the parallel composition operator, namely, we will write $a \times n$ to denote the parallel composition of n atomic elements a. Moreover, to simplify the counting mechanism, we might use different names for the same molecule when it belongs to different

[1] Kinetics are omitted from the table for simplicity.

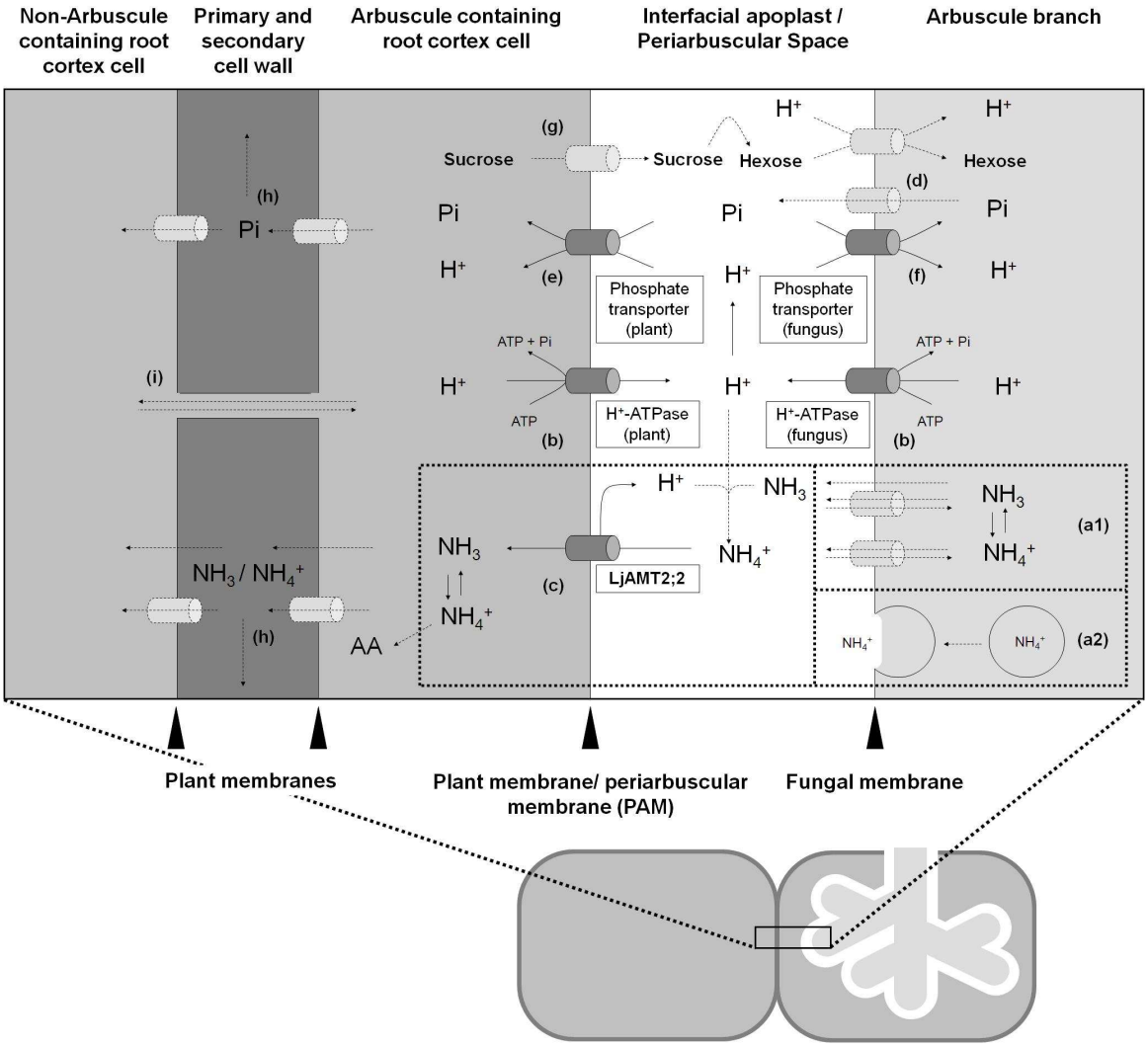


Figure 2: **(a1-2)** NH_3/NH_4^+ is released in the arbuscules from arginine which is transported from the extra- to the intraradical fungal structures [16]. NH_3/NH_4^+ is released by so far unknown mechanisms (transporter, diffusion **(a1)** or vesicle-mediated **(a2)**) into the periarbuscular space (PAS) where, due to the acidic environment, its ratio shifts towards NH_4^+ ($> 99.99\%$). **(b)** The acidity of the interfacial apoplast is established by plant and fungal H^+-ATPases [21, 5] thus providing the energy for H^+-dependent transport processes. **(c)** The NH_4^+ ion is deprotonated prior to its transport across the plant membrane via the LjAMT2;2 protein and released in its uncharged NH_3 form into the plant cytoplasm. The NH_3/NH_4^+ acquired by the plant is either transported into adjacent cells or immediately incorporated into amino acids (AA). **(d)** Phosphate is released by so far unknown transporters into the interfacial apoplast. **(e)** The uptake of phosphate on the plant side then is mediated by mycorrhiza-specific Pi-transporters [25, 18]. **(f)** AM fungi might control the net Pi-release by their own Pi-transporters which may reacquire phosphate from the periarbuscular space [5]. **(g)** Plant derived carbon is released into the PAS probably as sucrose and then cleaved into hexoses by sucrose synthases [22] or invertases [39]. AM fungi then acquire hexoses [41, 42] and transport them over their membrane by so far unknown hexose transporters. It is likely that these transporters are proton co-transporter as the GpMST1 described for the glomeromycotan fungus Geosiphon pyriformis [31]. Exchange of nutrients between arbusculated cells and non-colonized cortical cells can occur by apoplastic **(h)** or symplastic **(i)** ways.

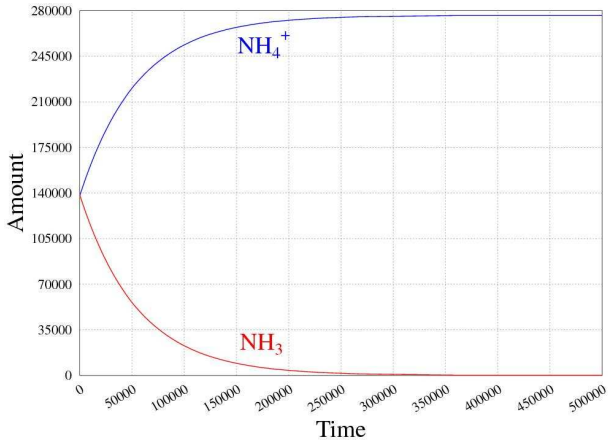


Figure 3: Extracellular equilibrium between NH_3 and NH_4^+.

compartments. For instance, occurrences of the molecule NH_3 inside the plant cell are called NH_3_inside.

3.1 NH_3/NH_4^+ Equilibrium

We decided to start modelling a simplified pH equilibrium, at the interspatial level (right part of section **(c)** in Figure 2), without considering H_2O, H^+ and OH^-; therefore we tuned the reaction rates in order to reach the correct percentages of NH_3 over total NH_3/NH_4^+ in the different compartments. Like these all the rates and initial terms used in this work are obtained by manual adjustments made looking at the simulations results and trying to keep simulations times acceptable - we plan to refine these rates and numbers in future works to reflect more deeply the available biological data. Following [18], we consider an extracellular pH of 4.5 [19]. In such conditions, the percentage of molecules of NH_3 over the sum $NH_3 + NH_4^+$ should be around 0.002. The reaction we considered is the following:

$$NH_3 \underset{k_2}{\overset{k_1}{\rightleftharpoons}} NH_4^+$$

with $k_1 = 0.018 * 10^{-3}$ and $k_2 = 0.562 * 10^{-9}$. One can translate this reaction with the SCLS rules.

$$NH_3 \overset{k_1}{\mapsto} NH_4^+ \tag{R1}$$

$$NH_4^+ \overset{k_2}{\mapsto} NH_3 \tag{R2}$$

In Figure 3 we show the results of this first simulation given the initial term $T = NH_3 \times 138238 \,|\, NH_4^+ \times 138238$.

This equilibrium is different at the intracellular level (pH around 7 and 8) [15], so we use two new rules to model the transformations of NH_3 and NH_4^+ inside the cell, namely:

$$NH_3\_\text{inside} \overset{k_1}{\mapsto} NH_4^+\_\text{inside} \tag{R3}$$

$$NH_4^+\_\text{inside} \overset{k_2'}{\mapsto} NH_3\_\text{inside} \tag{R4}$$

where $k_2' = 0.562 * 10^{-6}$.

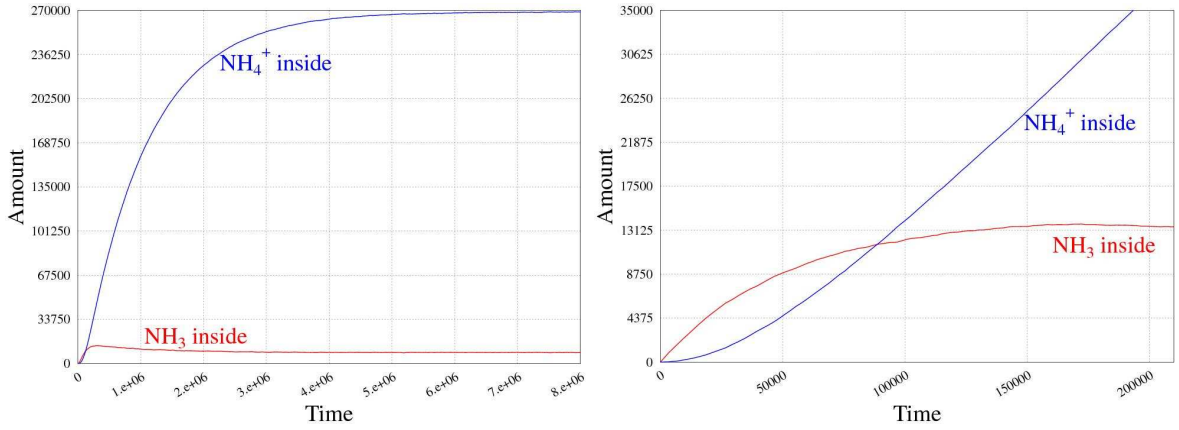


Figure 4: At high NH_4^+ concentration.

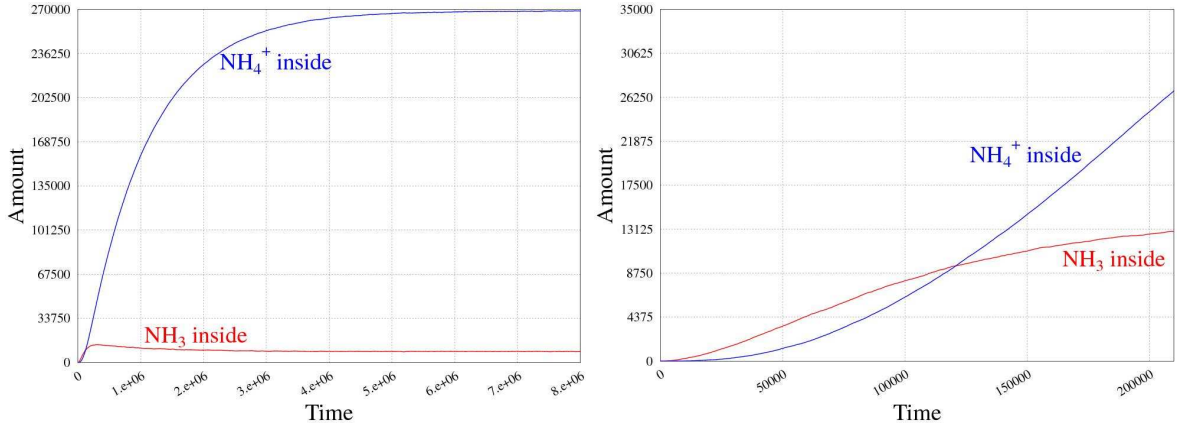


Figure 5: At low NH_4^+ concentration.

3.2 LjAMT2;2 Uptake

We can now present the SCLS model of the uptake of the LjAMT2;2 transporter (left part of section **(c)** in Figure 2). We add a looping sequence modelling an arbusculated plant cell. Since we are only interested in the work done by the LjAMT2;2 transporter, we consider a membrane containing this single element. The work of the transporter is modelled by the rule:

$$NH_4^+ \mid (LjAMT2)^L \rfloor (X) \overset{k_t}{\mapsto} H^+ \mid (LjAMT2)^L \rfloor (X \mid NH_3\_\text{inside}) \tag{R5}$$

where $k_t = 0.1 * 10^{-5}$.

We can investigate the uptake rate of the transporter at different initial concentrations of NH_3 and NH_4^+. Figure 4 and Figure 5 show the results for the initial terms
$T_1 = NH_3 \times 776 \mid NH_4^+ \times 276400 \mid (LjAMT2)^L \rfloor \varepsilon$ and $T_2 = NH_3 \times 276400 \mid NH_4^+ \times 776 \mid (LjAMT2)^L \rfloor \varepsilon$, respectively (the left one represents the whole simulations, while on the right there is a magnification of their initial segment).

We can also investigate the uptake rate of the transporter at different extracellular pH. Namely, we consider an extracellular pH equal to the intracellular one (pH around 7 and 8), obtained by imposing

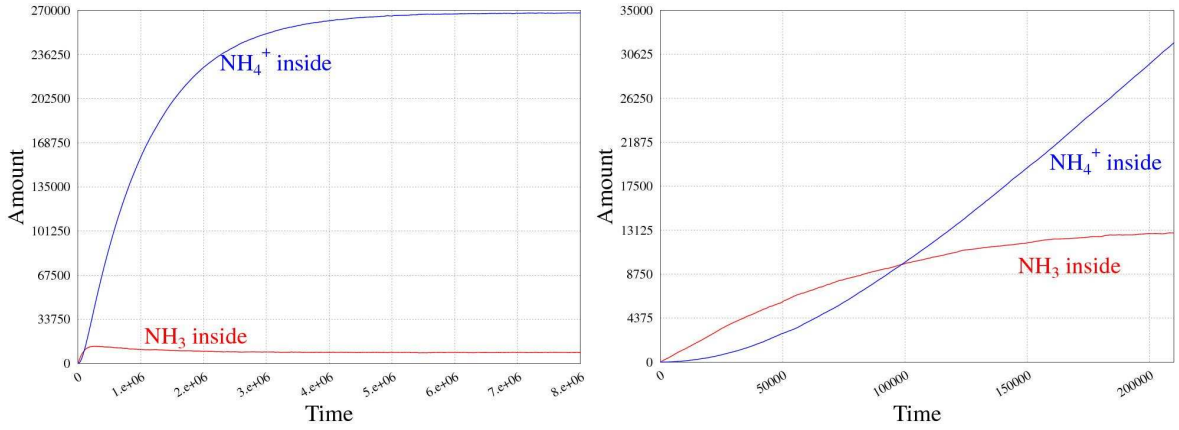


Figure 6: At extracellular pH=7.

$R1$ and $R2$ equal to $R3$ and $R4$, respectively, i.e. $k_2 = k'_2$. Figure 6 shows the results for the initial terms $T_1 = NH_3 \times 138238 \,|\, NH_4^+ \times 138238 \,|\, (LjAMT2)^L \rfloor \varepsilon$.

Since now we modeled the transporter supposing that no active form of energy is required to do the actual work - which means that the NH_4^+ gradient between the cell and the extracellular ambient is sufficient to determine a net uptake. The predicted tridimensional structure of LjAMT2;2 suggests that it does not use ATP [2] as an energy source [18], nevertheless trying to model an "energy consumption" scenario is interesting to make some comparisons. Since this is only a proof of concept there is no need to specify here in which form this energy is going to be provided, furthermore as long as we are only interested in comparing the initial rates of uptake we can avoid defining rules that regenerate energy in the cell. Therefore, rule R5 modelling the transporter role can be modified as follows:

$$NH_4^+ \,|\, (LjAMT2)^L \rfloor (ENERGY \,|\, X) \overset{k'_t}{\mapsto} H^+ \,|\, (LjAMT2)^L \rfloor (X \,|\, NH_3\_\text{inside}) \qquad \text{(R5')}$$

which consumes an element of energy within the cell. We also make this reaction slower, since it is now catalysed by the concentration of the $ENERGY$ element, actually, we set $k_{t'} = 0.1 * 10^{-10}$. Given the initial term $T = NH_3 \times 138238 \,|\, NH_4^+ \times 138238 \,|\, (LjAMT2)^L \rfloor (ENERGY \times 100000)$ we obtain the simulation result in Figure 7. Note that the uptake work of the transporter terminates when the $ENERGY$ inside the cell is completely exhausted.

3.3 NH_4^+ Diffusing from the Fungus

We now model the diffusion of NH_4^+ from the fungus to the extracellular level (sections **(a1)**, and **(a2)** of Figure 2). In section **(a1)** of the figure, the passage of NH_4^+ to the interfacial periarbuscular space happens by diffusion. We can model this phenomenon by adding a new compartment, representing the fungus, from which NH_4^+ flows towards the fungus-plant interface. This could be modelled through the rule:

$$(FungMembr)^L \rfloor (NH_4^+ \,|\, X) \overset{k_f}{\mapsto} NH_4^+ \,|\, (FungMembr)^L \rfloor X \qquad \text{(R6)}$$

By varying the value of the rate k_f one might model different externalization speeds and thus test different hypotheses about the underlying mechanism. In Figure 8 we give the simulation result, with

[2] ATP is the "molecular unit of currency" of intracellular energy transfer [26] and is used by many transporters that work against chemical gradients.

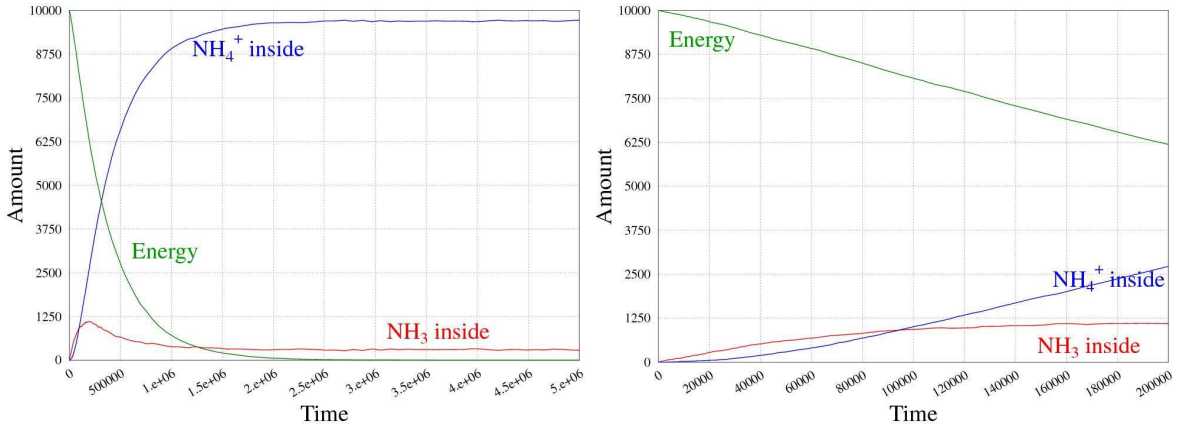


Figure 7: LjAMT2;2 with active energy.

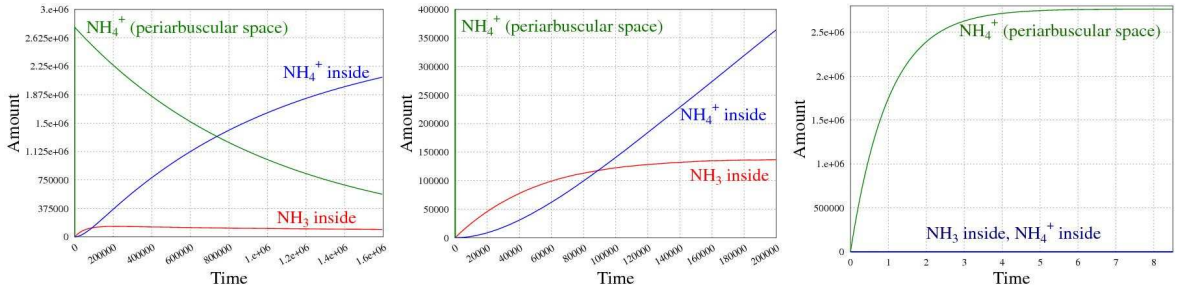


Figure 8: Diffusing NH_4^+ from the fungus, $k_f = 1$.

three different magnification levels, going from the whole simulation on the left to the very first seconds on the right, obtained from the initial term $T_f = (FungMembr)^L \rfloor (NH_4^+ \times 2764677) \mid (LjAMT2)^L \rfloor \varepsilon$ with $k_f = 1$. In the initial part, one can see how fast, in this case, NH_4^+ diffuses into the periarbuscular space (in the figure, $NH4$ represents the quantity of NH_4^+ in that part of the system). In Figure 9 we give the simulation result obtained from the same initial term T_f with a slower diffusion rate, namely $k_f = 0.01 * 10^{-3}$ (note that the magnification levels in the three panels are different with respect to those in Figure 8).

Finally, we would like to remark, without going into the simulation details, how we can model in a rather natural way the portion **(a2)** of Figure 2 in SCLS. Namely, we need some rules to produce vesicle containing NH_4^+ molecules within the fungal cell. Once the vesicle is formed, another rule drives its exocytosis towards the interfacial space, and thus the diffusion of the previously encapsulated NH_4^+ molecules. The needed rules are given in the following:

$$(FungMembr)^L \rfloor (NH_4^+ \mid X) \overset{k_c}{\mapsto} (FungMembr)^L \rfloor (X \mid (Vesicle)^L \rfloor NH_4^+) \tag{R7}$$

$$(FungMembr)^L \rfloor (NH_4^+ \mid Y \mid (Vesicle)^L \rfloor X) \overset{k_a}{\mapsto} (FungMembr)^L \rfloor (Y \mid (Vesicle)^L \rfloor (NH_4^+ \mid X)) \tag{R8}$$

$$(FungMembr)^L \rfloor (Y \mid (Vesicle)^L \rfloor X) \overset{k_e}{\mapsto} X \mid (FungMembr)^L \rfloor Y \tag{R9}$$

Where rule R7 models the creation of a vesicle, rule R8 model the encapsulation of an NH_4^+ molecule within the vesicle and rule R9 models the exocytosis of the vesicle content.

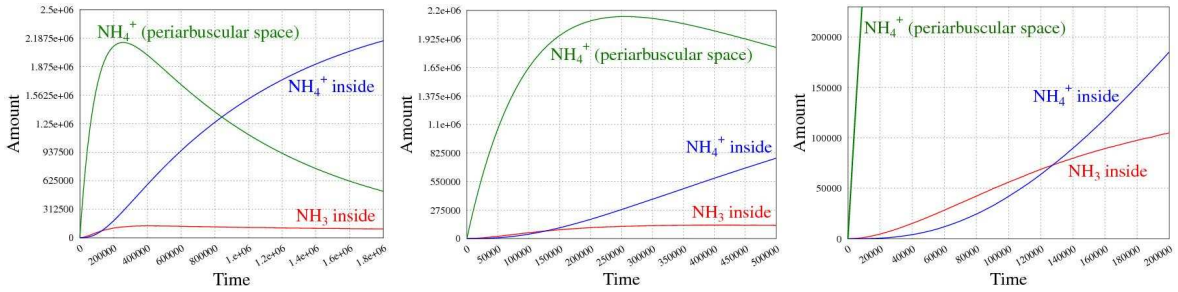


Figure 9: Diffusing NH_4^+ from the fungus, $k_f = 0.01 * 10^{-3}$.

4 Discussion

We dissected the route for the passage of NH_3 / NH_4^+ from the fungus to the plant in known and hypothetical mechanisms which were transformed in rules. Further, also the properties of the different compartments and their influence on the transported molecules were included, thus giving a first model for the simulation of the nutrients transfer. With the model so far we can simulate the behaviour of the system when varying parameters as the different compartments pH, the initial substrate concentrations, the transport/diffusion speeds and the energy supply.

We can start comparing the two simulations with the plant cell with the LjAMT2;2 transporter placed in different extracellular situations: low NH_4^+ concentration (Figure 5) or high NH_4^+ concentration (Figure 4). As a natural consequence of the greater concentration, the ammonium uptake is faster when the simulation starts with more NH_4^+, as long as the LjAMT2;2 can readily import it. The real situation should be similar to this simulation, assuming that the level of extracellular NH_4^+ / NH_3 is stable, meaning an active symbiosis.

The simulation which represents an extracellular pH around 7 (Figure 6) shows a decreased internalisation speed with respect to the simulation in Figure 4, as could be inferred from the concentrations of NH4_inside and NH3_inside in the plots on the right (focusing on the initial activity): this supports experimental data about the pH-dependent activity of the transporter and suggests that the extracellular pH is fundamental to achieve a sufficient ammonium uptake for the plants. It's noticeable how the initial uptake rate in this case is higher, despite the neutral pH, than the rate obtained considering an "energy quantum" used by the transporter (which has the same starting term), as could be seen in the right panels of Figure 7 and Figure 6. These results could enforce the biological hypothesis that, instead of ATP, a NH_4^+ concentration gradient (possibly created by the fungus) is used as energy source by the LjAMT2;2 protein.

The simulations which also consider the fungal counterpart are interesting because they provide an initial investigation of this rather poorly characterized side of the symbiosis and confirm that plants can efficiently gain ammonium if NH_4^+ is released from the fungi. This evidence supports the last biological hypothesis about how fungi supply nitrogen to plants [16, 10], and could lead to further models which could suggest which is the needed rate for NH_4^+ transport from fungi to the interfacial apoplast; thus driving biologists toward one (or some) of the nowadays considered hypotheses (active transport of NH_4^+, vesicle formation, etc.).

5 Conclusions and Future Work

This paper reports on the use of SCSL to simulate and understand some biological behaviour which is still unclear to biologists, and it also constitues a first attempt to predict biological behaviour and give some directions to biologists for future experiments.

SCLS has been particularly suitable to model the symbiosis (where substances flow through different cells) thanks to its feature to model compartments, and their membranes, in a simple and natural way. Our simulations have confirmed some of the latest experimental results about the LjAMT2;2 transporter [18] and also support some of the hypotheses about the energy source for the transport. These are the first steps towards a complete simulation of the symbiosis and open some interesting paths that could be followed to better understand the nutrient exchange.

As demonstrated by heterologous complementation experiments in yeast, mycorrhiza-specific plant transporters [20, 18] show different uptake efficiencies under varying pH conditions. Looking on this and the results from the simulations with different pH conditions in the periarbuscular space new experiments for a determination of the uptake kinetics (K_m-values) under a range of pH seems mandatory. Another conclusion from the model is that, for accurate simulation, exact in vivo concentration measurements have to be carried out even though at the moment this is a difficult task due to technical limitations.

As shown by various studies, many transporters on the plant side show a strong transcriptionally regulation and the majority of them are thought to be localized at the plant-fungus interface [17]. Consequently a quantification of these proteins in the membrane will be a prerequisite for an accurate future model.

It is known that some of these transporters (e.g. PT4 phosphate transporters) obtain energy from proton gradients established by proton pumps (Figure 2) whereas others as the LjAMT2;2 and aquaporines [14] use unknown energy supplies or are simply facilitators of diffusion events along gradients. Thus they conserve the membranes electrochemical potential for the before mentioned proton dependent transport processes. To make the story even more complex some of these transporters which are known to be regulated in the AM symbiosis (e.g. aquaporines) show overlaps in the selectivity of their substrates [23, 24, 43, 31]. Consequently integration of interrelated transporters in a future model will be a necessary and challenging task. It is quite probable that also transporters for other macronutrients (potassium and sulfate) which might be localized in the periarbuscular membrane influence the electrochemical gradients for the known transport processes.

With the ongoing sequencing work [27] on the arbuscular model fungus *Glomus intraradices* transporters on the fungal side of the symbiotic compartment are likely to be identified and characterized soon. Data from such future experiments could be integrated in the model and help to answer the question whether transporter mediated diffusion or vesicle based excretion events lead to the release of ammonium into the periarbuscular space [10].

Further questions about the plant nutrient uptake and competitive fungal reimport process [10, 5] might be answered. Based on the transport properties of orthologous transporters from different fungal and/or plant species, theories could be developed which explain different mycorrhiza responsiveness of host plants; meaning why certain plant-AM fungus combinations have a rather disadvantageous than beneficial effect for the plant.

In future research on AM and membrane transport processes in general values from measurements of concentrations or kinetics can and have to be included in the model and will show how the whole system is influenced by these values. Vice versa simulations could be the base for new hypotheses and experiments. As a future extension of the modelling technique, we plan to follow the direction taken in [12, 4] using type systems to guarantee that an SCLS term satisfies certain biological properties (enforced

by a set of typing rules), and also investigate how type systems could enrich the study of quantitative systems.

Acknowledgements We thank the referees for their insightful comments. The final version of the paper improved due to their suggestions.

References

[1] A. Abate, Y. Bai, N. Sznajder, C. Talcott, and A. Tiwari. Quantitative and probabilistic modeling in Pathway Logic. In *IEEE 7th International Symposium on Bioinformatics and Bioengineering*. IEEE, 2007.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Science, 2007.

[3] R. Alur, C. Belta, and F. Ivancic. Hybrid modeling and simulation of biomolecular networks. In *HSCC*, volume 2034 of *LNCS*, pages 19–32. Springer, 2001.

[4] B. Aman, M. Dezani-Ciancaglini, and A. Troina. Type disciplines for analysing biologically relevant properties. *Electr. Notes Theor. Comput. Sci.*, 227:97–111, 2009.

[5] R. Balestrini, J. Gomez-Ariza, L. Lanfranco, and P. Bonfante. Laser microdissection reveals that transcripts for five plant and one fungal phosphate transporter genes are contemporaneously present in arbusculated cells. *Mol.Plant Microbe Interact.*, 20:1055–1062, 2007.

[6] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, P. Tiberi, and A. Troina. Stochastic calculus of looping sequences for the modelling and simulation of cellular pathways. *Transactions on Computational Systems Biology*, IX:86–113, 2008.

[7] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. Bisimulation congruences in the calculus of looping sequences. In *ICTAC*, volume 4281 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2006.

[8] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. A calculus of looping sequences for modelling microbiological systems. *Fundam. Inform.*, 72(1-3):21–35, 2006.

[9] L. Cardelli. Brane calculi. In *CMSB*, volume 3082 of *LNCS*, pages 257–278. Springer, 2005.

[10] M. Chalot, D. Blaudez, and A. Brun. Ammonia: a candidate for nitrogen transfer at the mycorrhizal interface. *Trends in Plant Science*, 11:263–266, 2005.

[11] V. Danos and C. Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.

[12] M. Dezani-Ciancaglini, P. Giannini, and A. Troina. A type system for required/excluded elements in CLS. In *DCM*, page To appear, 2009.

[13] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977.

[14] T. Gonen and T. Walz. The structure of aquaporins. *Q.Rev.Biophys.*, 39:361–396, 2006.

[15] E. Gout, R. Bligny, and R. Douce. Regulation of intracellular pH values in higher plant cells. Carbon-13 and phosphorus-31 nuclear magnetic resonance studies. *J. Biol. Chem.*, 267:13903–13909, 1992.

[16] M. Govindarajulu, P.E. Pfeffer, H. Jin, J. Abubaker, D.D. Douds, J.W. Allen J.W., H. Bucking, P.J. Lammers, and Y. Shachar-Hill. Nitrogen transfer in the arbuscular mycorrhizal symbiosis. *Nature*, 435:819–823, 2005.

[17] M. Guether, R. Balestrini, M.A. Hannah, M.K. Udvardi, and P. Bonfante. Genome-wide reprogramming of regulatory networks, transport, cell wall and membrane biogenesis during arbuscular mycorrhizal symbiosis in *Lotus japonicus*. *New Phytol.*, 182:200–212, 2009.

[18] M. Guether, B. Neuhauser, R. Balestrini, M. Dynowski, U. Ludewig, and P. Bonfante. A mycorrhizal-specific ammonium transporter from *Lotus japonicus* acquires nitrogen released by arbuscular mycorrhizal fungi. *Plant Physiology*, 150:73–83, 2009.

[19] M. Guttenberger. Arbuscules of vesicular-arbuscular mycorrhizal fungi inhabit an acidic compartment within plant roots. *Planta*, 211:299–304, 2000.

[20] M.J. Harrison, G.R. Dewbre, and J. Liu. A phosphate transporter from *medicago truncatula* involved in the acquisition of phosphate released by arbuscular mycorrhizal fungi. *Plant Cell.*, 14:2413–2429, 2002.

[21] B. Hause and T. Fester. Molecular and cell biology of arbuscular mycorrhizal symbiosis. *Planta*, 221:184–196, 2005.

[22] N. Hohnjec, A.M. Perlick, A. Puhler, and H. Kuster. The *Medicago truncatula* sucrose synthase gene Mt-SucS1 is activated both in the infected region of root nodules and in the cortex of roots colonized by arbuscular mycorrhizal fungi. *Mol.Plant Microbe Interact.*, 16:903–915, 2003.

[23] L.M. Holm, T.P. Jahn, A.L. Moller, J.K. Schjoerring, D. Ferri, D.A. Klaerke, and T. Zeuthen. NH_3 and NH_4^+ permeability in aquaporin-expressing xenopus oocytes. *Pflugers Arch.*, 450:415–428, 2005.

[24] T.P. Jahn, A.L. Moller, T. Zeuthen, L.M. Holm, D.A. Klaerke, B. Mohsin, W. Kuhlbrandt, and J.K. Schjoerring. Aquaporin homologues in plants and mammals transport ammonia. *FEBS Lett.*, 574:31–36, 2004.

[25] H. Javot, N. Pumplin, and M.J. Harrison. Phosphate in the arbuscular mycorrhizal symbiosis: transport properties and regulatory roles. *Plant Cell and Environment*, 30:310–322, 2007.

[26] JR. Knowles. Enzyme-catalyzed phosphoryl transfer reactions. *Annu. Rev. Biochem.*, 49:877–919, 1980.

[27] F. Martin, V. Gianinazzi-Pearson, M. Hijri, P. Lammers, N. Requena, I.R. Sanders, Y. Shachar-Hill, H. Shapiro, G.A. Tuskan, and J.P.W Young. The long hard road to a completed *Glomus intraradices* genome. *New Phytologist*, 180, 2008.

[28] H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid Petri net representation of gene regulatory network. In *Prooceedings of Pacific Symposium on Biocomputing*, pages 341–352. World Scientific Press, 2000.

[29] P. Milazzo. *Qualitative and quantitative formal modelling of biological systems*. PhD thesis, University of Pisa, 2007.

[30] J.C. Mills, K.A. Roth, R.L. Cagan, and J.I. Gordon. DNA microarrays and beyond: completing the journey from tissue to cell. *Nat Cell Biol.*, 3(8):943, 2001.

[31] C.M. Niemietz and S.D. Tyerman. Channel-mediated permeation of ammonia gas through the peribacteroid membrane of soybean nodules. *FEBS Lett.*, 465:110–114, 2000.

[32] M. Parniske. Arbuscular mycorrhiza: the mother of plant root endosymbioses. *Nat. Rev. Microbiol.*, 6:763–775, 2008.

[33] C. Priami. Stochastic pi-calculus. *Comput. J.*, 38(7):578–589, 1995.

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

[35] G. Păun. *Membrane computing. An introduction*. Springer, 2002.

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

[37] A. Regev and E. Shapiro. *The π-calculus as an abstraction for biomolecular systems*, pages 219–266. Natural Computing Series. Springer, 2004.

[38] G. Scatena. Development of a stochastic simulator for biological systems based on the calculus of looping sequences. Master's thesis, University of Pisa, 2007.

[39] S. Schaarschmidt, T. Roitsch, and B. Hause. Arbuscular mycorrhiza induces gene expression of the apoplastic invertase LIN6 in tomato (lycopersicon esculentum) roots. *J.Exp.Bot.*, 57:4015–4023, 2006.

[40] M. Schena, R.A. Heller, T.P. Theriault, K. Konrad, E. Lachenmeier, and R.W. Davis. Microarrays: biotechnology's discovery platform for functional genomics. *Trends Biotechnol.*, 17(6):217–218, 1999.

[41] Y. Shacharhill, P.E. Pfeffer, D. Douds, S.F. Osman, L.W. Doner, and R.G. Ratcliffe. Partitioning of intermediary carbon metabolism in vesicular-arbuscular mycorrhizal leek. *Plant Physiology*, 108:7–15, 1995.

[42] M.D.Z. Solaiman and M. Saito. Use of sugars by intraradical hyphae of arbuscular mycorrhizal fungi revealed by radiorespirometry. *New Phytologist*, 136:533–538, 1997.

[43] S.D. Tyerman, C.M Niemietz, and H. Bramley. Plant aquaporins: multifunctional water and solute channels with expanding roles. *Plant Cell Environ.*, 25:173–194, 2002.

Quantifying the implicit process flow abstraction in SBGN-PD diagrams with Bio-PEPA

Laurence Loewe[1] Stuart Moodie[2,1] Jane Hillston[2,1]

[1]Centre for System Biology at Edinburgh
[2]School of Informatics
The University of Edinburgh
Scotland

Laurence.Loewe@ed.ac.uk Stuart.Moodie@ed.ac.uk Jane.Hillston@ed.ac.uk

For a long time biologists have used visual representations of biochemical networks to gain a quick overview of important structural properties. Recently SBGN, the Systems Biology Graphical Notation, has been developed to standardise the way in which such graphical maps are drawn in order to facilitate the exchange of information. Its qualitative Process Diagrams (SBGN-PD) are based on an implicit Process Flow Abstraction (PFA) that can also be used to construct quantitative representations, which can be used for automated analyses of the system. Here we explicitly describe the PFA that underpins SBGN-PD and define attributes for SBGN-PD glyphs that make it possible to capture the quantitative details of a biochemical reaction network. We implemented SBGNtext2BioPEPA, a tool that demonstrates how such quantitative details can be used to automatically generate working Bio-PEPA code from a textual representation of SBGN-PD that we developed. Bio-PEPA is a process algebra that was designed for implementing quantitative models of concurrent biochemical reaction systems. We use this approach to compute the expected delay between input and output using deterministic and stochastic simulations of the MAPK signal transduction cascade. The scheme developed here is general and can be easily adapted to other output formalisms.

1 Introduction

To describe biological pathway information visually has many advantages and Systems Biology Graphical Notation Process Diagrams (SBGN-PD [19]) have been developed for this purpose. Like electronic circuit diagrams, they aim to unambiguously describe the structure of a complex network of interactions using graphical symbols. To achieve this requires both a collection of symbols and rules for their valid combination. SBGN-PD is a visual language with a precise grammar that builds on an underlying abstraction as the basis of its semantics (see p.40 [19]). We call this underlying abstraction for SBGN-PD the "Process Flow Abstraction" (PFA). It describes biological pathways in terms of processes that transform elements of the pathway from one form into another. The usefulness of an SBGN-PD description critically depends on the faithfulness of the underlying PFA and a tight link between the PFA and the glyphs used in diagrams. The graphical nature of SBGN-PD allows only for qualitative descriptions of biological pathways. However, the underlying PFA is more powerful and also forms the basis for quantitative descriptions that could be used for analysis. Such descriptions, however, need to allow the inclusion of the corresponding mathematical details like parameters and equations for computing the frequency with which reactions occur.

Here we aim to make explicit the PFA that already underlies SBGN-PD implicitly. This serves a twofold purpose. First, a better and more intuitive understanding of the underlying abstraction will make it easier for biologists to construct SBGN-PD diagrams. Second, the PFA is easily quantified and making it explicit can facilitate the quantitative description of SBGN-PD diagrams. Such descriptions can then

R.J.Back, I.Petre, E. de Vink (Eds.): Computational Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 93–107, doi:10.4204/EPTCS.6.7

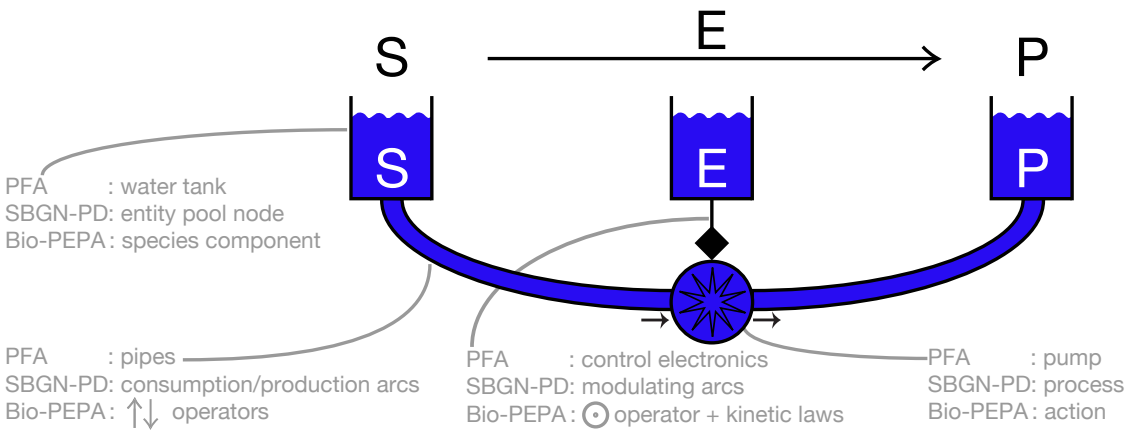


Figure 1: An overview of the process flow abstraction. The chemical reaction at the top is translated into an analogy of water tanks, pipes and pumps that can be used to visualise the process flow abstraction. The various elements are also mapped into SBGN-PD and Bio-PEPA terminology.

be used directly for predicting quantitative properties of the system in simulations. Here we demonstrate how this could work by mapping SBGN-PD to a quantitative analysis system. We use the process algebra Bio-PEPA [5, 1] as an example, but our mapping can be easily applied to other formalisms as well.

The rest of the paper is structured as follows. First we provide an overview of the implicit PFA with the help of an analogy to a system of water tanks, pipes and pumps (Section 2). In Section 3 we explain how this system can be extended in order to capture quantitative details of the PFA. We then show how SBGN-PD glyphs can be mapped to a quantitative analysis framework, using the Bio-PEPA modelling environment [1] as an example (Section 4). In Section 5 we discuss various internal mechanisms and data structures needed for translation into any quantitative analysis framework. As an example of how the translation process works, we apply our new translation tool "SBGNtext2BioPEPA" [20, 21] to a simple model of the MAPK signalling cascade [15], which we automatically translate into Bio-PEPA, where we analyse the stochastic behaviour of the time needed for the cascade to be switched from "off" to "on". We end by reviewing related work and providing some perspectives for further developments.

2 The implicit Process Flow Abstraction of SBGN-PD

The PFA behind SBGN-PD is best introduced in terms of an analogy to a system of many water tanks that are connected by pipes. Each pipe either leads to or comes from a pump whose activity is regulated by dedicated electronics. In the analogy, the water is moved between the various tanks by the pumps. In a biochemical reaction system, this corresponds to the biomass that is transformed from one chemical species into another by chemical reactions. SBGN-PD aims to also allow for descriptions at levels above individual chemical reactions. Therefore the water tanks or chemical species are termed "entities" and the pumps or chemical reactions are termed "processes". For an overview, see Figure 1. We now discuss the correlations between the various elements in the analogy and in SBGN-PD in more detail. In this discussion we occasionally allude to SBGNtext, which is a full textual representation of the semantics of SBGN-PD (developed to facilitate automated translation of SBGN-PD into other formalisms; see [20, 21]). Here are the key elements of the PFA:

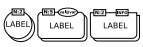
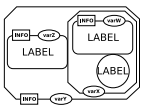
SBGN-PD glyph	EPNType	class type	comment
	Unspecified	material	EPN with unknown specifics
	SimpleChemical	material	EPN
	Macromolecule	material	EPN
	NucleicAcidFeature	material	EPN
	-	material	EPN multimer specified by cardinality
	Complex	container	EPN; arbitrary nesting allowed
	Source	conceptual	external source of molecules
	Sink	conceptual	removal from the system
	PerturbingAgent	conceptual	external influence on a reaction

Table 1: Categories of "water tanks" in the PFA correspond to types of entity pool nodes in SBGN-PD. The complex and the multimers are shown with exemplary auxiliary units that specify cardinality, potential chemical modifications and other information.

Water tanks = entity pool nodes (EPNs). Each water tank stands for a different pool of entities, where the amount of water in a tank represents the biomass that is bound in all entities of that particular type that exist in the system. Typical examples for such pools of identical entities are chemical species like metabolites or proteins. SBGN-PD does not distinguish individual molecules within pools of entities, as long as they are within the same compartment and identical in all other important properties. An overview of all types of EPNs (i.e. categories of water tanks) in SBGN-PD is given in Table 1. To unambiguously identify an entity pool in SBGNtext and in the code produced for quantitative analysis, each entity pool is given a unique `EntityPoolNodeID`. The PFA does not conceptually distinguish between non-composed entities and entities that are complexes of other entities. Despite potentially huge differences in complexity they are all "water tanks" and further quantitative treatment does not treat them differently.

Pipes = consumption and production arcs. Pipes allow the transfer of water from one tank to another. Similarly, to move biomass from one entity pool to another requires the consumption and production of entities as symbolised by the corresponding arcs in SBGN-PD (see Table 3, page 101). These arcs connect exactly one process and one EPN. The thickness of the pipes could be taken to reflect stoichiometry, which is the only explicit quantitative property that is an integral part of SBGN-PD. Production arcs take on a special role in reversible processes by allowing for bidirectional flow.

Pumps = processes. Pumps move water through the pipes from one tank to another. Similarly, processes transform biomass bound in one entity to biomass bound in another entity, i.e., processes transform one entity into another. The speed of the pump in the analogy corresponds to the frequency with which the reaction occurs and determines the amount of water (or biomass) that is transported between tanks (or that is converted from one entity to another, respectively). Processes can belong to different types in SBGN-PD (Table 2) and are unambiguously identified by a unique

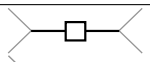
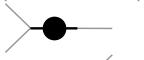
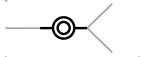
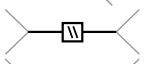
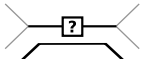
SBGN-PD glyph	ProcessType	meaning
	Process	normal known processes
	Association	special process that builds complexes
	Dissociation	special process that dissolves complexes
	Omitted	several known processes are abstracted
	Uncertain	existence of this process is not clear
	Observable	this process is easily observable

Table 2: Categories of "pumps" in the PFA correspond to types of processes in SBGN-PD.

`ProcessNodeID` in SBGNtext. This allows arcs to clearly define which process they belong to and by finding all its arcs, each process can also identify all EPNs it is connected to.

Reversible processes. SBGN-PD allows for processes to be reversible if they are symmetrically modulated (p.28 [19]). Thus, there may be flows in two directions, however the net flow at any given time will be unidirectional. The PFA does not prescribe how to implement this. For simplicity, our analogy assumes pumps to be unidirectional, like many real-world pumps. Thus bidirectional processes in our analogy are represented as two pumps with corresponding sets of pipes and opposite directions of flow. In our implementation we follow SBGN-PD in separating the left hand side and right hand side of reversible processes for an unambiguous description of reality if all relevant arc glyphs look like production arcs (p.32 [19]). For more details, see [21].

Control electronics for pumps = modulating arcs and logic gates. In the analogy, pumps need to be regulated, especially in complex settings. This is achieved by control electronics. In SBGN-PD, the same is done by various types of modulation arcs, logic arcs and logic gates [19]. They all contribute to determining the frequency of the reaction. Since SBGN-PD does not quantify these interactions, most of our extensions for quantifying SBGN-PD address this aspect. Each arc connects a "water tank" with a given `EntityPoolNodeID` and a "pump" with a given `ProcessNodeID`. Ordinary modulating arcs can be of type `Modulation` (most generic influence on reaction), `Stimulation` (catalysis or positive allosteric regulation), `Catalysis` (special case of stimulation, where activation energy is lowered), `Inhibition` (competitive or allosteric) or `NecessaryStimulation` (process is only possible, if the stimulation is "active", i.e. has surpassed some threshold). The glyphs are shown in Table 3 (page 101), where their mapping to Bio-PEPA is discussed. One might misread SBGN-PD to suggest that Consumption/Production arcs cannot modulate the frequency of a process. However, kinetic laws frequently depend on the concentration of reactants, implying that these arcs can also contribute to the "control electronics" (e.g. report "level of water in tank"). Another part of the "control electronics" are *logical operators*. These simplify modelling, when a biological function can be approximated by a simple on/off logic that can be represented by boolean operators. SBGN-PD supports this simplification by providing the logical operators "AND", "OR" and "NOT", which are connected by "logic arcs" with the rest of the diagram (logic arcs convert to and from the non-boolean world).

Groups of water tanks = compartments, submaps and more. The PFA is complete with all the elements presented above. However, to make SBGN-PD more useful for modelling in a biological context, SBGN-PD has several features that make it easier for biologists to recognise various subsets of entities that are related to each other. For example, entities that belong to the same compartment can be grouped together in the compartment glyph and functionally related entities can be placed on the same submap. In the analogy, this corresponds to grouping related water tanks together. SBGN-PD also supports sophisticated ways for highlighting the inner similarities between entities based on a knowledge of their chemical structure (e.g. modification of a residue, formation of a complex). Stretching the analogy, this corresponds to a way of highlighting some similarities between different water tanks. None of these groupings are important for the PFA in principle or for quantitative analysis, as long as different "water tanks" remain separate.

3 Extensions for quantitative analysis

The process flow abstraction that is implicit in all SBGN process diagrams can be used as a basis to quantify the systems they describe. After we made the PFA explicit above, we now discuss the attributes that need to be added to the various SBGN-PD glyphs in order to allow for automatic translation of SBGN-PD diagrams into quantitative models. These attributes are stored as strings in SBGNtext (our textual representation of SBGN-PD, see [21]) and are attached to the corresponding glyphs by a graphical SBGN-PD editor. They do not require a visual representation that compromises the visual ease-of-use that SBGN-PD aims for. Next we discuss the various attributes that are necessary for the glyphs of SBGN-PD to support quantitative analysis. We do not discuss auxiliary units, submaps, tags and equivalence arcs here, as they do not require extensions for supporting quantitative analysis.

3.1 Quantitative extensions of EntityPoolNodes

For quantitative analysis, each unique EPN requires an `InitialMoleculeCount` to unambiguously define how many entities exist in this pool in the starting state. We followed developments in the SBML standard in using counts of molecules instead of concentrations, since SBGN-PD also allows for multiple compartments, which makes the use of concentrations very cumbersome (see section 4.13.6, p.71f. in [16]). For entities of type `Perturbation`, the `InitialMoleculeCount` is interpreted as the numerical value associated with the perturbation, even though its technical meaning is not a count of molecules. Entities of the type `Source` or `Sink` are both assumed to be effectively infinite, so `InitialMoleculeCount` does not have a meaning for these entities. Beyond a unique `EntityPoolNodeID` and `InitialMoleculeCount`, no other information on entities is required for quantitative analysis.

3.2 Quantitative extensions of Arcs

Arcs link entities and processes by storing their respective IDs and the `ArcType`. The simplest arcs are of type `Consumption` or `Production` and do not require numerical information beyond the stoichiometry that is already defined in SBGN-PD as a property of arcs that can be displayed visually in standard SBGN-PD editors. Logic arcs will be discussed below. All modulating arcs are part of the "control electronics" and affect the frequency with which a process happens. They link to EPNs to inform the process about the presence of enzymes, for example. Modulation is usually governed by parameters or other important quantities for the given process (e.g. Michaelis-Menten-constant).

To make the practical encoding of a model easier, we define process parameters that conceptually belong to a particular modulating entity as a list of `QuantitativeProperties` in the arc pointing to that entity. This is equivalent to seeing the set of parameters of a reaction as something that is specific to the interaction between a particular modulator and the process it modulates. Other approaches are also possible, but lead to less elegant implementations. Storing parameters in equations requires frequent and possibly error-prone changes (e.g. many different Michaelis-Menten equations). One could also argue that the catalytic features are a property of the enzyme and thus make parameters part of EPNs; however this either forces all Michaelis-Menten reactions of an enzyme to happen at the same speed or requires cumbersome naming conventions to manage different affinities for different substrates.

To refer to parameters we specify the `ManualEquationArcID` of an arc and then the name of the parameter that is stored in the list of `QuantitativeProperties` of that arc. This scheme reduces clutter by limiting the scope of the relevant namespace (only few arcs per process exist, so `ManualEquationArcID`s only need to be unique within that immediate neighbourhood). Thus parameter names can be brief, since they only need to be unique within the arc. The `ManualEquationArcID` is specified by the user in the visual SBGN-PD editor and differs from ArcID, a globally unique identifier that is automatically generated by the graphical editor. The `ManualEquationArcID` allows for user-defined generic names that are easy to remember, such as 'Km' and 'vm' for Michaelis-Menten reactions. It should be easily accessible within the graphical editor, just as the parameters that are stored within an arc.

Logical operators and logic arcs. To facilitate the use of logical operators in quantitative analyses one needs to convert the integer molecule counts of the involved EPNs to binary signals amenable to boolean logic. Thus SBGNtext supports "incoming logic arcs" that connect a "source entity" or "source logical result" with a "destination logic operator" and apply an "input threshold" to decide whether the source is above the threshold ("On") or below the threshold ("Off"). To this end, a graphical editor needs to support the "input threshold" as a numerical attribute that the user can enter; all other information recorded in incoming logic arcs is already part of an SBGN diagram. Once all signals are boolean, they can be processed by one or several logical operators, until the result of this operation is given in the form of either 0 ("Off") or 1 ("On"). This result then needs to be converted back to an integer or float value that can be further processed to compute process frequencies. Thus a graphical editor needs to support corresponding attributes for defining a low and a high output level.

3.3 Quantitative extensions of ProcessNodes

For quantitative analyses, a ProcessNode must have a unique name and an equation that computes the propensity, which is proportional to the probability that this process occurs next, based on the current global state of the model. Since the `ProcessType` is not required for quantitative analyses, it does not matter whether a process is an ordinary `Process`, an `Uncertain` process or an `Observable` process, for example. For all these ProcessNodes, graphical editors need to support attributes for the manual specification of a `ProcessNodeID`, and a `PropensityFunction`. These attributes are then stored in SBGNtext. If support for bidirectional processes is desired, then graphical editors need to facilitate entering a propensity function for the backward process as well. Propensity functions compute the propensity of a unidirectional process to be the next event in the model and can be used directly by simulation algorithms and solvers [12].

To instantiate the propensity function, a translator needs to replace all aliases by their true identity. We use the following syntax for a parameter alias that is substituted by the actual numeric value (or a globally defined parameter) from the corresponding arc:

<par: ManualEquationArcID.QuantitativePropertyName>

While translating to Bio-PEPA this would be simply substituted with a corresponding parameter name. The parameter is then defined elsewhere in the Bio-PEPA code to have the numerical value stored in the corresponding property of the arc. To allow the numerical analysis tool to access an EPN count at runtime we replace the following entity alias by the EntityPoolNodeID that the corresponding arc links to:

<ent: ManualEquationArcID >

This is shorter than the EntityPoolNodeID and allows the reuse of propensity functions if kinetic laws are identical and the manual IDs follow the same pattern. It is desirable that there is no need to specify the EntityPoolNodeID. It is fairly long and generated automatically to reflect various properties that make it unique. It would be cumbersome to refer to in the equation and it would require a mechanism to access the automatically generated EntityPoolNodeID before a SBGNtext file is generated. Also any changes to an entity that would affect its EntityPoolNodeID would then also require a change in all corresponding propensity functions, a potentially error-prone process. The same substitution mechanism can be used to provide access to properties of compartments (see [21]).

In addition to these aliases, functions use the typical standard arithmetic rules and operators that are merely handed through to the analysis tool.

4 Mapping SBGN-PD elements to Bio-PEPA

In this section we explain how to use the semantics of SBGN-PD to map a SBGN-PD model to a formalism for the quantitative analysis of biochemical systems. We are using Bio-PEPA as an example, but our approach is general and can be applied to many other formalisms.

4.1 The Bio-PEPA language

Bio-PEPA is a stochastic process algebra which models biochemical pathways as interactions of distinct entities representing reactions of chemical species [5, 1]. A process algebra model captures the behaviour of a system as the actions and interactions between a number of entities, where the latter are often termed "processes", "agents" or "components". In PEPA and Bio-PEPA these are built up from simple *sequential components* [5, 14, 3]. Different process algebras support different modelling styles for biochemical systems [3]. Stochastic process algebras, such as PEPA [14] or the stochastic π-calculus [24], associate a random variable with each action to represent the mean of its exponentially distributed waiting time. In the stochastic π-calculus, interactions are strictly binary whereas in Bio-PEPA the more general multiway synchronisation is supported. The syntax of Bio-PEPA is defined as [5] :

$$S ::= (\alpha,\kappa)\ \mathsf{op}\ S \mid S + S \mid C \qquad\qquad P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(x)$$

where S is a sequential *species component* that represents a chemical species (termed "process" in some other process algebras and "EntityPoolNode" in SBGN-PD), C is a constant pointing to an S, P is a *model component* that describes the set $\mathcal{L}$ of possible interactions between species components (these "interactions" or "actions" correspond to "processes" in SBGN-PD and can represent chemical reactions). A count of molecules or a concentration of S is given by $x \in \mathbb{R}_0^+$. In the prefix term "$(\alpha,\kappa)\ \mathsf{op}\ S$", κ is the *stoichiometry coefficient* and the operator op indicates the role of the species in the reaction α.

Specifically, op $= \downarrow$ denotes a *reactant*, $\uparrow$ a *product*, $\oplus$ an *activator*, $\ominus$ an *inhibitor* and $\odot$ a generic *modifier*, which indicates more complex roles than $\oplus$ or $\ominus$. The operator "+" expresses a choice between possible actions. Finally, the process $P \underset{\mathcal{L}}{\bowtie} Q$ denotes the synchronisation between components: the set $\mathcal{L}$ determines those activities on which the operands are forced to synchronise. When $\mathcal{L}$ is the set of common actions, we use the shorthand notation $P \bowtie Q$.

A Bio-PEPA system $\mathcal{P}$ is defined as a 6-tuple $\langle \overset{*}{\mathcal{V}}, \mathcal{N}, \mathcal{K}, \mathcal{F}_R, Comp, P \rangle$, where: $\mathcal{V}$ is the set of compartments, $\mathcal{N}$ is the set of quantities describing each species (includes the initial concentration), $\mathcal{K}$ is the set of all parameters referenced elsewhere, $\mathcal{F}_R$ is the set of functional rates that define all required kinetic laws, *Comp* is the set of species components S that highlight the reactions an entity can take part in and P is the system model component. Bio-PEPA models (i) represent reversible reactions as pairs of irreversible forward and backward reactions, (ii) treat the same species in different states or compartments as different species represented by distinct Bio-PEPA components and (iii) assume static compartments. See [5] for more details.

A variety of analysis techniques can be applied to a single Bio-PEPA model, facilitating the easy validation of analysis results when the analyses address the same issues [2] and enhancing insight when the analyses are complementary [4]. Currently supported analysis techniques include stochastic simulation at the molecular level, ordinary differential equations, probabilistic model checking and numerical analysis of continuous time Markov chains [5, 1, 9].

4.2 SBGN-PD mapping

Here we map the core elements of SBGN-PD to Bio-PEPA (see [20] for an implementation).

Entity Pool Nodes Due to the rich encoding of information in the `EntityPoolNodeID`, Bio-PEPA can treat each distinct `EntityPoolNodeID` as a distinct species component. This removes the need to explicitly consider any other aspects such as entity type, modifications, complex structures and compartments, as all such information is implicitly passed on to Bio-PEPA by using the `EntityPoolNodeID` as the name for the corresponding species component. To define the set $\mathcal{N}$ of a Bio-PEPA system requires the attribute `InitialMoleculeCount` for each EPN (see Section 3).

Processes All SBGN-PD `ProcessTypes` are simply represented as reactions in Bio-PEPA. Compiling the corresponding set $\mathcal{F}_R$ relies on the attribute `PropensityFunction` and a substitution mechanism that makes it easy to define these functions manually. To help humans understand references to processes in the sets $\mathcal{F}_R$ and *Comp* requires recognizable names for SBGN-PD `ProcessNodeID`s that map directly to their identifiers in Bio-PEPA. Thus graphical editors need to allow for manual `ProcessNodeID`s.

Reversible processes. The translator supports reversible SBGN-PD processes by dividing them into two unidirectional processes for Bio-PEPA. The translator reuses the manually assigned `ProcessNodeID` and augments it by "_F" for forward reactions and "_B" for backward reactions. These two unidirectional processes are then treated independently. When compiling the species components in Bio-PEPA, every time a `LeftHandSide` arc is found, the translator assumes that the corresponding forward and backward processes have been defined and will augment the process name by "_F" for forward reactions and "_B" for backward reactions, while adding the corresponding Bio-PEPA operator for reactant and product. `RightHandSide` arcs are handled in the same way. Thus the production arc glyph in SBGN-PD has three distinct meanings as shown in Table 3.

Arcs. The arcs in SBGN-PD define which entities interact in which processes. Thus arcs play a pivotal role in defining the species components in Bio-PEPA. Since arcs can store kinetic parameters, they are also important for defining parameters in Bio-PEPA. As kinetic law definitions in Bio-PEPA frequently refer to such parameters, we use the ArcID that is automatically generated by the graphical

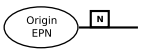
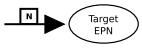
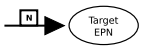
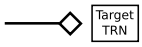
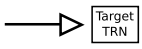
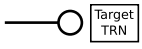
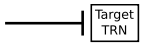
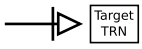
SBGN-PD glyph	ArcType	Bio-PEPA symbol	Bio-PEPA code
	Consumption	↓	<<
	Production	↑	>>
	LeftHandSide	↓ and ↑	<< and >>
	RightHandSide	↑ and ↓	>> and <<
	Modulation	⊙	(.)
	Stimulation	⊕	(+)
	Catalysis	⊕	(+)
	Inhibition	⊖	(-)
	NecessaryStimulation	⊙	(.)

Table 3: "Water pipes and control electronics": Mapping arcs between entities and processes in SBGN-PD to operators in Bio-PEPA species components. The "symbols" are the formal syntax of Bio-PEPA, while the "code" gives the concrete syntax used in the Bio-PEPA Eclipse Plug-in [1].

editor to substitute the local manual arc references in propensity functions by globally unique parameters names (see Section 3). The type of an arc indicates both the role of the connected entity in the process (consumed reactant, product or modifier of rate) and the chemical nature of the reaction (catalysis, stimulation, inhibition, necessary stimulation or the most generic modification). Thus the type of an arc can be mapped directly to the operator "op" described in the Bio-PEPA syntax shown in Table 3.

Logical operators Logical operators require the conversion of integer molecule counts of the relevant EPNs to binary signals and after some boolean logic processing back to low and high integer values. As evident from the implementation scheme above, the use of all quantitative properties culminates in the correct formulation of the corresponding propensity functions that determine the probability that the corresponding process will be the next to occur. Thus an implementation of logical operators requires that their results be included in the corresponding propensity functions. The current scheme of implementing propensity functions relies heavily on substituting the various components into the final equation, so that Bio-PEPA will ultimately only see one formula per propensity function. In this context the implementation of logical operators requires the insertion of a formula in the propensity function that computes the result of the boolean operations from their integer input. An arbitrarily complex logic operator network can be constructed from the following basic building blocks:

- *Convert from integers or double floats to boolean values*. This is best done by a specially defined mathematical function that takes an integer or float signal and compares it to a specified threshold, giving back either 0 (signal < threshold) or 1 (signal ≥ threshold). The definition of such a function is not complicated and is implemented in the current Bio-PEPA Eclipse Plug-in (starting with version 0.1).
- AND operator. Multiply all boolean inputs to arrive at the output.
- NOT operator. The arithmetic expression (1- *input*) computes the *output*
- OR operator. Sum all inputs (0/1) and test if it is greater than 1 using the threshold function.

Of these, only the threshold function is not widely available, even if it is easy to implement.

5 Converter implementation and internal representation

We chose Java as implementation language for the converter described above, due to the good portability of the resulting binaries and the use of Java in the Bio-PEPA Eclipse Plug-in [1]. We defined a grammar for SBGNtext in the Extended Backus-Naur-Form (EBNF) as supported by ANTLR [22], which automatically compiles the Java sources for the corresponding parser that stores all important parsing results in a number of coherently organised internal TreeMaps. To compile a working Bio-PEPA model three main loops over these TreeMaps are necessary: over all entities, over all processes and over all parameters. To illustrate the translation we refer to "`code`" examples from Figure 2.

The **loop over all entities** (e.g. "`m_MAPKK_PP`") compiles the species components as well as the model description required by Bio-PEPA. The latter is a list of all participating `EntityPoolNodeID`s combined by the cooperation operator "$< * >$" that automatically synchronises all common actions. This simplification depends on all processes in SBGN-PD having unique names and fixed lists of reactants with no mutually exclusive alternatives in them. The first condition can be enforced by the tools that produce the code, the second is ensured by the reaction-style of describing processes in SBGN-PD.

For each species component a loop over all arcs finds the arcs that are connected to it (e.g. "`st27`") and that store all relevant `ProcessNodeID`s (e.g. "`K_P_act`"). The same loop determines the respective role of the component (as reflected by the choice of the Bio-PEPA operator in Table 3; e.g. "`(+)`").

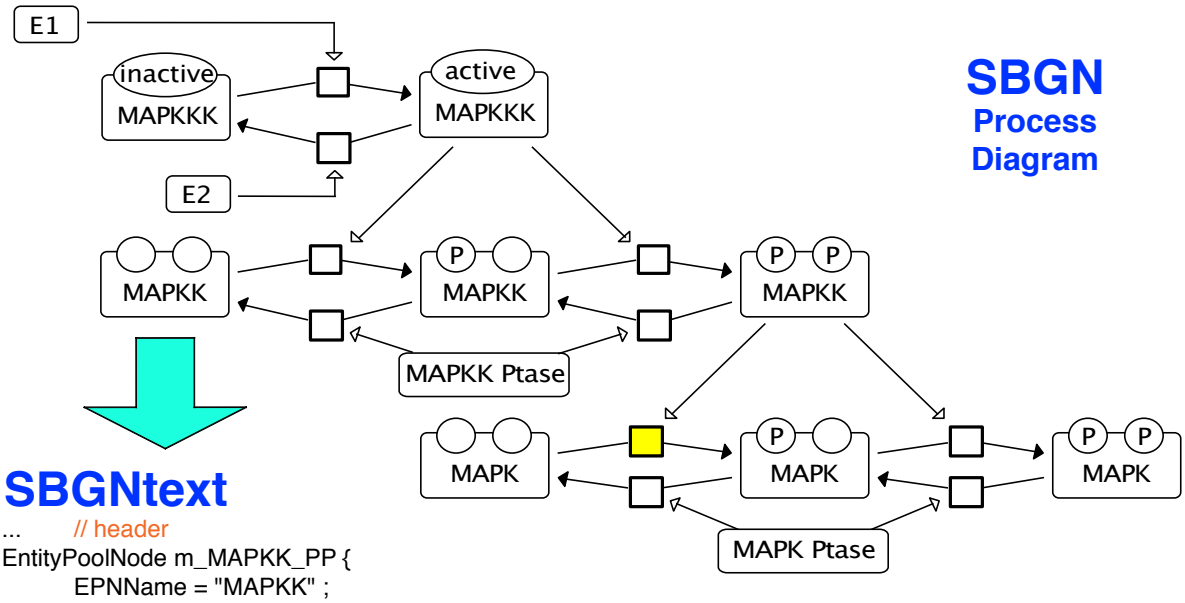
The **loop over all processes** (e.g. "`K_P_act`") compiles the kinetic laws by substituting aliases (e.g. "`<par: enz.kcat>`" or "`<ent: enz>`") for parameters ("`st27_kcat`") and EPNs ("`m_MAPKK_PP`") in the propensity functions specified in the graphical editor. Each function is handled separately by a dedicated function parser that queries the TreeMaps generated when parsing the SBGNtext file.

The **loop over all quantitative properties** of the model defines the parameters in Bio-PEPA (e.g. "`st27_kcat`"). It is possible to avoid this step by inserting the direct numerical values into the equations processed in the second loop. However, this substantially reduces the readability of equations in the Bio-PEPA code and makes it difficult for third party tools to assist in the automated generation of parameter combinations. Thus we defined a scheme that automatically generates parameter names to maximise the readability of equations (combine ArcID "`st27`" and name of the quantitative property "`kcat`").

To facilitate walking over the various collections specified above, the TreeMaps are organised in four sets, one for entities, one for processes, one for arcs and one for quantitative properties. Each of these sets is characterised by a common key to all maps within the set. This facilitates the retrieval of related parse products for the same key from a different map. Since all this information is accessible from Java code, it is easily conceivable to use the sources produced in this work for reading SBGNtext files in a wide variety of contexts. The system is easy to deploy, since it involves few files and the highly portable ANTLR runtime library. Our converter is called SBGNtext2BioPEPA and sources are available [20].

6 Example: Stochasticity in the MAPK cascade

Here we illustrate our translation by applying SBGNtext2BioPEPA [20] to a real-world example. We implemented a simple model of the Mitogen-Activated Protein Kinase (MAPK) signal transduction cascade [18, 15] in order to investigate the time needed for the cascade to switch from "off" to "on". The general pattern of the MAPK cascade is well conserved across many biological taxa and triggers a highly varied range of molecular responses [15]. Such a broad conservation suggests not only important functionality (maintained by purifying selection), but also an astonishing flexibility in how a MAPK cascade might be implemented (it has to work in a wide range of contexts). For example, MAPK cascades operate in



SBGNtext

```
...      // header
EntityPoolNode m_MAPKK_PP {
        EPNName = "MAPKK" ;
        EPNType =  Macromolecule  ;
        EPNState = { ( Label = "P" ) ( Label = "P" ) } ;
        InitialMoleculeCount =  0 ;
} ...    // 11 more EPNs
ProcessNode K_P_act {
        ProcessType = Process;
        PropensityFunction =
          "<par: enz.kcat>*<ent: enz>*<ent: subs>
        / (<par: enz.Km_CountCell>*<ent: subs>)" ;
} ...    // 9 more ProcessNodes
Arc st27 {
        ManualEquationArcID = enz ;
        ArcType = Stimulation ;
        Entity = m_MAPKK_PP  ;
        Process = K_P_act  ;
        Stoichiometry = 1 ;
        QuantitativeProperties = {
                ( Km_CountCell =   300.0  )
                ( kcat =  10.0  )
        } ;
} ...    // 29 more Arcs
...      // ending
```

Bio-PEPA

```
st27_Km_CountCell = 300;
st27_kcat = 10.0;
...      // 18 more parameters

kineticLawOf  K_P_act :
        st27_kcat  *
        m_MAPKK_PP  *  m_MAPK_P   /
        ( st27_Km_CountCell  *  m_MAPK_P )
;  ...   // 9 more kinetic laws

m_MAPKK_PP =
        ( K_act          , 1 ) (+) m_MAPKK_PP  +
        ( K_P_act        , 1 ) (+) m_MAPKK_PP  +
        ( KK_P_act       , 1 ) >> m_MAPKK_PP  +
        ( KK_PP_deact    , 1 ) << m_MAPKK_PP
;  ...   // 11 more species components

// model component with initial counts per cell
m_MAPKK_PP [0] <*> m_MAPK_P [0] <*> ...
   // 9 more species components [initial count]
```

Figure 2: A possible SBGN-PD representation of a MAPK cascade with fragments from the corresponding SBGNtext code and the automatically generated Bio-PEPA code. Most biologists would find it far more natural to draw the diagram at the top than to write the Bio-PEPA code. The code excerpts focus on EPN MAPK-PP and the reaction it catalyses (`'K_P_act'`, yellow node). All parameters are scaled to represent counts of molecules per cell (including the Michaelis-Menten constants). For the full code and newer versions see [20]. Biologically, the input signal is a change in `E1`, the output signal a change in `MAPK_PP`. The back reactions on each of the three levels can be thought of as a "conveyer belt" that constantly turns active molecules (on the right) into passive ones (on the left). `E1` must overcome the conveyer belt on the first level for sequentially overcoming the other conveyer belts too. In many such systems either active or inactive molecules are essentially absent.

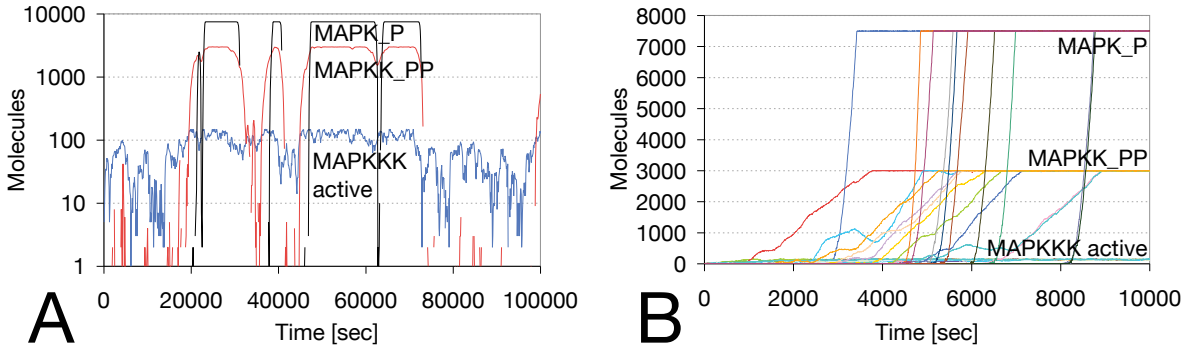


Figure 3: Noise in the MAPK cascade. (A) One stochastic simulation trace, where E1 = E2 = 20 makes the cascade switch between the "on" and "off"state, as activating and deactivating reactions at the first level are equally strong. (B) Ten stochastic simulation traces, where E1 = 21, E2 = 20. If activating reactions are only slightly stronger than deactivating reactions at the first level of the cascade, the cascade can be expected to switch "on". However the time until the "on" state is reached is subject to considerable stochasticity, which is mostly determined by stochasticity during the initial stages of the switching processes.

oocytes of the frog *Xenopus* and in yeast cells, despite their 5 million fold differences in size [15, 27] and substantial differences in kinetic constants [15, 17].

Figure 2 shows a SBGN-PD diagram of the basic structure of a MAPK cascade. It accepts "input" in the form of a change in the concentration of the enzyme E1 and then propagates the signal by changing the concentrations of the various intermediate substrates and enzymes until the "output" enzyme MAPK_PP is affected. If the input is off (low E1), the output is off too (no MAPK_PP). If the input is on (E1 above threshold), output is reliably on as well (high MAPK_PP). Important properties for signalling cascades include the reliability of transmitting a signal and the speed with which this happens. Building on the model in [15], a recent study explored the expected percentage of active output and the time until all output is activated [18].

Here we automatically translate our SBGNtext model of the MAPK cascade via SBGNtext2BioPEPA into a Bio-PEPA model (see [20] for code), which we analyse with the Bio-PEPA Eclipse Plug-in [1]. We report the results of stochastic simulations using the Gibson-Bruck algorithm [11, 12]. Each reaction is assumed to follow Michaelis-Menten kinetics, resulting in the following propensity function:

$$\frac{k_{cat}[count/sec] * Enzyme[count] * Substrate[count]}{K_M[count] + Substrate[count]}$$

where [count] indicates the absolute number of molecules of this entity within the cell, k_{cat} denotes the number of substrate molecules that can be processed by one enzyme at maximal speed, K_M denotes the Michaelis-Menten constant that is an inverse measure of the affinity between enzyme and substrate. We assume $K_M = 300$ and $k_{cat} = 10$ for all reactions (see [15, 17] for other possible values). All our simulations start with molecule counts of E2 = 20, MAPKKK = 150, MAPKK = 3000, MAPKK_Ptase = 100, MAPK = 7500,MAPK_Ptase = 2000 and 0 for all other entities, reflecting the equilibrium "off" state. These molecule counts roughly reflect MAPK cascades in yeast, and are about 6 orders of magnitude below corresponding counts in *Xenopus* oocytes [15]. We tested various E1 input signals and found the following results that cannot be obtained without translating SBGN-PD into a quantitative formalism.

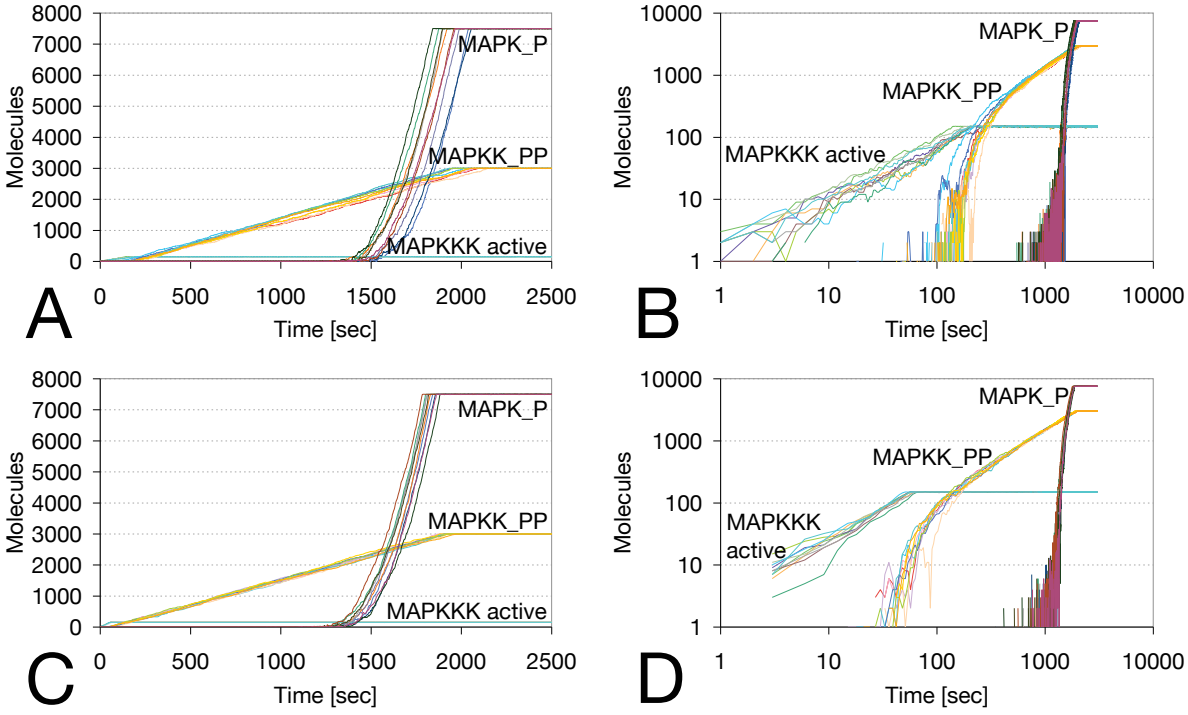


Figure 4: Increasing the number of E1 molecules from 40 (upper part, A, B) to 100 (lower part, C, D) reduces the stochasticity of the time to the completion of the switch (see linear plots on the left, A, C). The stochasticity in the initial stages is considerable in both cases for this system (see log plots on the right, B, D). Ten stochastic simulation traces are shown. E2 = 20; for other parameters, see text.

1. The cascade remains "off", if E1<E2 (not shown).
2. The cascade switches beween "on" and "off", if E1 = E2 as shown in Figure 3A.
3. The cascade switches reliably to "on" if E1>E2, but at E1 = 21 there is considerable noise in the "signalling time" (from switching E1 "on" until all output MAPK_PP is switched "on" as well; see Figure 3B).
4. Further increasing E1 substantially reduces the variability in signalling times and slightly improves speed (Figure 4).
5. Additional tests have shown that the overall signalling speed is very strongly influenced by k_{cat} (not shown).

7 Related work

There are various tools that map visual diagrams to quantitative modelling environments (e.g. SPiM [23], BlenX [7], kappa [6], Snoopy [13], EPN-PEPA [26], JDesigner [25]). However the corresponding graphical notations are not as rich as SBGN-PD and are thus not easily applied to the wide range of

scenarios that SBGN-PD was designed for. Since SBGN-PD is emerging as a new standard, it is clearly desirable to translate from SBGN-PD to a quantitative environment.

Since the first draft of SBGN-PD has been published in August 2008, a number of tools are being developed to support it. The graphical editor CellDesigner [10] supports a subset of SBGN-PD and can translate it into SBML which is supported by many quantitative analysis tools. However the process of adding quantitative information involves cumbersome manual interventions. This motivated work for SBMLsqueezer [8], a CellDesigner plug-in that supports the automatic construction of generalised mass action kinetics equations. While the automated suggestions for the kinetic laws from SBMLsqueezer might be of interest for some problems, the generated reactions contain many parameters that are extraordinarily difficult to estimate. Thus it is preferable to also allow the user to enter arbitrary kinetic laws that may have to be hand-crafted, but whose equations are simpler and require fewer parameter estimates. In SBGNtext2BioPEPA this is combined with mechanisms to reuse the code for such kinetic laws, greatly reducing practical difficulties and the potential for errors.

8 Conclusion and Perspectives

We have explicitly described the process flow abstraction that implicitly underlies SBGN-PD. We have used this abstraction to design a mechanism for translating SBGN-PD into code that can be used for quantitative analysis, using Bio-PEPA as an example. In order to do this we build on SBGNtext, a textual representation of SBGN-PD that we created [20, 21] and that focusses on the key functional SBGN-PD content, avoiding the clutter that comes from storing graphical details. We have developed our translator SBGNtext2BioPEPA in Java to facilitate its integration in the Systems Biology Software Infrastructure that is currently under development at the Centre for Systems Biology at Edinburgh (http://csbe.bio.ed.ac.uk/). SBGNtext2BioPEPA contains a parser for our SBGNtext format based on a formal ANTLR EBNF grammar and is freely available [20]. Building on the process flow abstraction and the internal representation of entities, processes, arcs and parameters in our code can make it easy to implement translations of SBGNtext to other modelling platforms. Since biologists are much more comfortable with drawing visual diagrams than writing code, support for translating SBGN-PD into quantitative analysis systems can play a key role in facilitating quantitative modelling.

Acknowledgements. We thank Stephen Gilmore for comments that improved this manuscript. The Centre for Systems Biology Edinburgh is a Centre for Integrative Systems Biology (CISB) funded by BBSRC and EPSRC, reference BB/D019621/1.

References

[1] Bio-PEPA homepage `http://www.biopepa.org/`. To install the Bio-PEPA Eclipse Plug-in by Adam Duguid follow the links from `http://homepages.inf.ed.ac.uk/jeh/Bio-PEPA/Tools.html` (2009)

[2] Calder M. , Duguid A., Gilmore S. and Hillston J.: Stronger computational modelling of signalling pathways using both continuous and discrete-space methods. Proc. of *CMSB'06*, LNCS vol. 4210, pp. 63–77 (2006)

[3] Calder, M. and Hillston, J.: Process algebra modelling styles for biomolecular processes. *Transactions on Computational Systems Biology*, in press, (2009)

[4] Ciocchetta F., Gilmore S., Guerriero M.-L. and Hillston J.: Stochastic Simulation and Probabilistic Model-Checking for the Analysis of Biochemical Systems. To appear in ENTCS (Proc. of PASM 2008).

[5] Ciocchetta F. and Hillston J.: A framework for the modelling and analysis of biological systems. Theoretical Computer Science, doi:10.1016/j.tcs.2009.02.037 (2009)

[6] Danos V., Feret J., Fontana W., Harmer R., Krivine J.: Rule-based modelling of cellular signalling. Lecture Notes in Computer Science, vol. 4703, pp.17-41, (2007).

[7] Dematté L., Priami C., Romanel A.: The BlenX Language: A Tutorial. SFM 2008, LNCS 5016:313-365, Springer (2008)

[8] Draerger A., Hassis N., Supper J., Schröder A., and Zell A.: SBMLsqueezer: A CellDesigner plug-in to generate kinetic rate equations for biochemical networks. BMC Systems Biology, 2:39 (2008)

[9] Duguid A., Gilmore S., Guerriero M.L., Hillston J. and Loewe L.: Design and development of software tools for Bio-PEPA. Proceedings of the 2009 Winter Simulation Conference (WSC'09), eds. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, Austin, Texas, to appear (2009).

[10] Funahashi, A.; Matsuoka, Y.; Jouraku, A.; Morohashi, M.; Kikuchi, N.; Kitano, H.: CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks, Proceedings of the IEEE vol. 96, issue 8, pp. 1254-1265, http://www.celldesigner.org/ (2008)

[11] Gibson M.A., Bruck J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. Journal of Physical Chemistry A, 104:1876-1889 (2000)

[12] Gillespie, D. T.: Stochastic simulation of chemical kinetics. Annu. Rev. Phys. Chem. 58:35-55 (2007)

[13] Heiner, M., Richter R., Schwarick M., Rohr C.: Snoopy - A tool to design and execute graph-based formalisms. Petri Net Newsletter 74 (April) ISSN 0931-1084, pp. 8-22, http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html (2008)

[14] Hillston, J.: *A Compositional Approach to Performance Modelling*, Cambridge University Press (1996)

[15] Huang C.Y., Ferrell J.E., Jr.: Ultrasensitivity in the mitogen-activated protein kinase cascade. Proc Natl Acad Sci USA, 93:10078-10083, (1996)

[16] Hucka M., et al.: Systems Biology Markup Language (SBML) Level 2 Version 4 Release 1. Nature Precedings http://dx.doi.org/10.1038/npre.2008.2715.1 and http://sbml.org/ (2008)

[17] Kholodenko, B.N.: Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. Eur J Biochem vol. 267 (6) pp. 1583-8 (2000)

[18] Kwiatkowska M., Norman G. and Parker D.: Using probabilistic model checking in systems biology. ACM SIGMETRICS Performance Evaluation Review vol. 35 (4) 14-21 (2008)

[19] Le Novère, N. et al.: Systems Biology Graphical Notation: Process Diagram Level 1. Nature Preceedings http://hdl.handle.net/10101/npre.2008.2320.1 (2008)

[20] Loewe L.: SBGNtext2BioPEPA. http://csbe.bio.ed.ac.uk/SBGNtext2BioPEPA/index.php (2009)

[21] Loewe L., Moodie S., Hillston J.: Technical Report: Defining a textual representation for SBGN Process Diagrams and translating it to Bio-PEPA for quantitative analysis of the MAPK signal transduction cascade. Technical Report of the School of Informatics, University of Edinburgh
http://csbe.bio.ed.ac.uk/SBGNtext2BioPEPA/index.php (2009)

[22] Parr, T.: The Definitive ANTLR Reference: Building Domain-Specific Languages. The Pragmatic Bookshelf, Raleigh, NC. http://www.antlr.org/ (2007)

[23] Phillips A.: A Visual Process Calculus for Biology. Jones and Bartlett Publishers, to appear (2009) http://research.microsoft.com/en-us/projects/spim/

[24] Priami C.: Stochastic π-calculus. *The Computer Journal*, 38(6), pp. 578–589, (1995)

[25] Sauro H.M., Hucka M., Finney A., Wellock C., Bolouri H., Doyle J. and Kitano H.: Next generation simulation tools: the Systems Biology Workbench and BioSPICE integration. OMICS. 2003 7(4):355-72, (2003). For the graphical front end JDesigner see: http://www.sys-bio.org/software/jdesigner.htm

[26] Shukla, A.: Mapping the Edinburgh Pathway Notation to the Performance Evaluation Process Algebra. MSc. Thesis, University of Trento (2007)

[27] Wallace R.A., Misulovin Z.: Long-term growth and differentiation of Xenopus oocytes in a defined medium. P Natl Acad Sci USA vol. 75 (11) pp. 5534-8 (1978)

Dynamical and Structural Modularity of Discrete Regulatory Networks

Heike Siebert
DFG Research Center MATHEON, Freie Universität Berlin
Arnimallee 6
D-14195 Berlin, Germany
siebert@mi.fu-berlin.de

A biological regulatory network can be modeled as a discrete function f that contains all available information on network component interactions. From f we can derive a graph representation of the network structure as well as of the dynamics of the system. In this paper we introduce a method to identify modules of the network that allow us to construct the behavior of f from the dynamics of the modules. Here, it proves useful to distinguish between dynamical and structural modules, and to define network modules combining aspects of both. As a key concept we establish the notion of symbolic steady state, which basically represents a set of states where the behavior of f is in some sense predictable, and which gives rise to suitable network modules. We apply the method to a regulatory network involved in T helper cell differentiation.

1 Introduction

Qualitative methods present a rigorous mathematical framework for modeling biological systems for which experimental data needed to determine kinetic parameters and mechanisms is lacking. The components of the system are modeled as variables adopting only finitely many values, so-called activity levels. In the simplest case, we obtain a Boolean representation, where the values 0 and 1 may for example represent a gene being inactive or active. In the general case, each component can have several activity levels, which may be appropriate depending on the biological data, and often is useful when modeling components that influence several other network components. A vector assigning each component an activity level then represents a state of the system. The information about network structure as well as the logical rules governing the behavior of the system in state space is represented by a discrete function f.

Although discrete networks are a strongly simplified representation of the original system, complex networks are hard to analyze, not least because the state space grows exponentially with the number of components. So, methods to reduce the complexity of the analysis are of great interest. One approach is to deconstruct the network in smaller building blocks that can be analyzed more easily, which leads to the notion of network modularity.

The idea of decomposing networks into modules is well-established in systems biology, although the notion of network module is not clear-cut. Often modules are defined based on biological criteria, that have to be translated into mathematical properties in order to identify them in a mathematical model (see e. g. [5, 4, 12]). Other approaches focus purely on the the graph representation of the network structure. Modules are defined as subgraphs satisfying graph theoretical characteristics often related to connectivity [3], or with statistical significance in comparison with random networks [11, 1]. In addition to this structural view, there are also approaches to find dynamical modules, see e. g. [6], that focus on identifying behavioral characteristics. However, in general the results obtained by analyzing such

R.J.Back, I.Petre, E. de Vink (Eds.): Computational
Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 109–124, doi:10.4204/EPTCS.6.8

modules in isolation do not translate to the original network, since additional influences have to be taken into account once the module is re-embedded in the original system. Here, the key is finding conditions that allow to draw conclusions about a complex network from knowledge obtained from module analysis, as e. g. possible in the modular response analysis approach in the context of metabolic networks and steady state fluxes [9, 2].

In this paper, we focus on the discrete modeling approach, presenting a method to identify *network modules* that allow us to derive precise information on the dynamics of the original system from the results of the analysis of the modules, building on ideas and significantly extending results from [17, 16]. In particular, we show that we can explicitly construct attractors of the original systems from network module attractors. Here, modularity is a key concept, and we exploit a purely structural as well as a purely dynamical view of modularity to eventually determine network modules combining important aspects of both. The core notion in our method is that of *symbolic steady state*. Such a state represents a set of constraints on the activity levels of the network components that allows us on the one hand to focus on dynamics restricted to subsets of state space, on the other hand enables us to identify dynamical and structural modules that render the basis for defining suitable network modules.

The paper is organized as follows. In the next section we describe the discrete modeling formalism we use throughout the paper, and introduce structural, dynamical and network modules. In Sect. 3 we establish the notion of symbolic steady state as well as related concepts. This is followed by the main results concerning network analysis utilizing modules in Sect. 4. We then illustrate the results for a class of networks, namely networks with input layer. In Sect. 6 we apply the method to the analysis of a regulator network involved in T helper cell differentiation proposed in [10]. We close with conclusions and perspectives.

2 Discrete regulatory networks

In this paper we model regulatory systems as discrete functions which capture all available information about network interactions and the logical rules governing the behavior of the system. Throughout the text, let us consider a system consisting of $n \in \mathbb{N}$ network components $\alpha_1, \ldots, \alpha_n$. In the following we identify a component α_i with its index i to simplify notation. Each component is interpreted as a variable which takes integer values that represent the different activity levels of the component. The literal meaning of those levels may be very different for different network components, for example they can represent levels of substance concentration, gene activity, presence or absence of a signal and so on. A vector assigning each component an activity level represents a state of the system, and the dynamics of the system is represented by state changes due to component interactions.

Definition 2.1 *For all $i \in \{1, \ldots, n\}$, let $p_i \in \mathbb{N}$, and set $X_i = \{0, 1, \ldots, p_i\}$. Set $X = X_1 \times \cdots \times X_n$, and let $f = (f_1, \ldots, f_n) : X \to X$ be a function. We call f a* network *comprising n components. For each $i \in \{1, \ldots, n\}$, the value p_i is the* maximal activity level *of α_i, and X_i is called the* range *of α_i. The set X is called the* state space *of f.*

Each coordinate function f_i of f describes the rules governing the behavior of the i-th network component depending on the state of the system. But f carries not only dynamical but also structural information on the system. Both aspects can be represented by directed graphs derived from f as we will see in the following two subsections. In the remainder of the paper f denotes a network as introduced in Def. 2.1

2.1 Structure

We represent the structure of a network by a signed directed (multi-)graph, where vertices represent the network components, and an edge from α_i to α_j signifies that the value of f_j depends on the activity level of α_i. The sign of the edge represents the character, i. e., activating or inhibiting, of the interaction. This description is inherently local in nature, so we first introduce a structural representation depending on the state of the system. This notion was introduced for Boolean functions in [13] and is used for multi-value functions in the form considered here in [14].

Definition 2.2 *Let $x \in X$. By $G(f)(x)$ we denote the directed signed (multi-)graph with vertex set $V = \{\alpha_1, \ldots, \alpha_n\}$ and edge set $E(x) \subseteq V \times V \times \{+,-\}$. An edge (i,j,ε) belongs to $E(x)$ iff there exists $c_i \in \{-1,+1\}$ such that $x_i + c_i \in X_i$ and*

$$\operatorname{sgn}\left(\frac{f_j((x_1,\ldots,x_{i-1},x_i+c_i,x_{i+1},\ldots,x_n)) - f_j(x)}{c_i}\right) = \varepsilon\,,$$

where the function $\operatorname{sgn} : \mathbb{Z} \to \{+,-,0\}$ *satisfies* $\operatorname{sgn}(0) = 0$, $\operatorname{sgn}(z) = +$ *if* $z > 0$, *and* $\operatorname{sgn}(z) = -$ *if* $z < 0$. *We call $G(f)(x)$ the* local interaction graph of f in x.

The local interaction graph in x is closely related to the discrete Jacobian matrix as introduced in [15] in the Boolean case. Note that in the multi-value other than in the Boolean case it is possible that $G(f)(x)$ contains parallel edges. There are at most two parallel edges from one vertex to another which then have opposite sign.

The local definition is easily extended, if we are interested in a representation of the interactions influencing the system behavior in larger subsets of state space.

Definition 2.3 *Let $Y \subseteq X$. We denote by $G(f)(Y)$ the union of the graphs $G(f)(x)$, $x \in Y$. We denote the graph $G(f)(X)$ also by $G(f)$ and call it the* global interaction graph of f.

In Fig. 1 (b) we see the global interaction graph of the network defined in Fig. 1 (a). The local interaction graph in the state $(1,1,0)$ is shown in Fig. 2 (a). To simplify notation we often write $G(x)$ and $G(Y)$ instead of $G(f)(x)$ and $G(f)(Y)$, respectively, if the corresponding function f is clear from the context.

When analyzing interaction graphs we are in particular interested in modules of the graph, a term for which there exists a variety of definitions as mentioned in the introduction. For our purposes it is convenient to use the term in the broadest sense, initially.

Definition 2.4 *A directed (multi-)graph $G' = (V^{G'}, E^{G'})$ is called a subgraph of a directed (multi-)graph $G = (V^G, E^G)$ if $V^{G'} \subseteq V^G$, $E^{G'} \subseteq E^G$, and every edge in $E^{G'}$ has both end-vertices in $V^{G'}$. We call a subgraph of $G(f)$* structural module of f.

Note that for a structural module the vertex set, the edge set or both may be smaller than for $G(f)$. For example, the graph shown in Fig. 1 (d) is a structural module of the function f given in the same figure. In general, local interaction graphs are structural modules of f. Of course, this definition is not very useful for analyzing the network structure or finding characteristics of the system. However, it is of conceptual advantage for us in the endeavor of defining network modules that combine structural and dynamical characteristics.

(a)

$f : \{0,1,2\}\times\{0,1\}\times\{0,1\} \to \{0,1,2\}\times\{0,1\}\times\{0,1\}$

$$
\begin{array}{lcl}
f_1(x) & = & 0, \text{ if } x_1 = x_2 = 0 \wedge x_3 = 1,\\
f_1(x) & = & 2, \text{ if } x_1 = 2 \wedge x_2 = 1,\\
f_1(x) & = & 1, \text{ otherwise,}\\
 & & \\
f_2(x) & = & 1, \text{ if } (x_1 \geq 1) \vee (x_2 = 1),\\
f_2(x) & = & 0, \text{ otherwise,}\\
 & & \\
f_3(x) & = & x_2 \wedge \neg x_3
\end{array}
$$

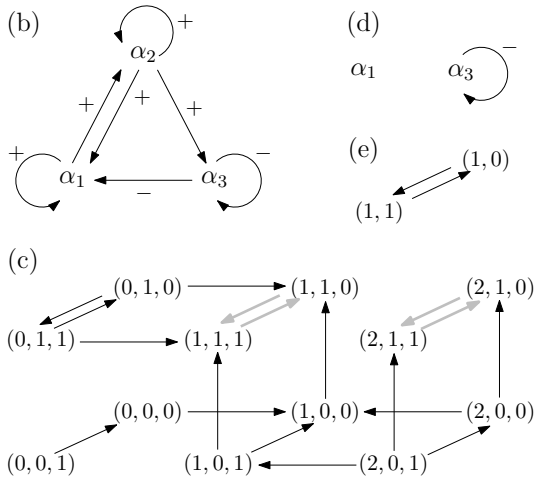


Figure 1: A network (a), its global interaction graph (b), and its state transition graph (c), where attractors are indicated by fat gray edges. Logical disjunction, conjunction and negation are represented by $\vee$, $\wedge$, and $\neg$, respectively. In (d) a structural, in (d) a dynamical module of f.

2.2 Dynamics

There are different approaches to deriving the dynamics of f. Commonly used is the so-called synchronous update strategy, where the successor of a state x is its image under f. A more realistic assumption is that not all changes indicated by differences in component values of x and $f(x)$ take the same amount of time to be executed, since they may represent very different biological processes. However, we lack the information to decide which of those processes of activity level change is the fastest. Therefore, all possible state transitions are taken into account leading to a non-deterministic representation of the dynamical behavior. Furthermore, we assume that a component value changes only by absolute value one in each transition, even though the function value may indicate a bigger change. This update method is called *asynchronous update* [18, 19].

Definition 2.5 *We denote by $S(f)$ the directed graph with vertex set X and edge set $E(S(f))$ defined as follows. An edge (x,x') is in $E(S(f))$ for states $x = (x_1,\ldots,x_n), x' = (x'_1,\ldots,x'_n) \in X$ if and only if $x' = f(x) = x$ or $x'_i = x_i + \mathrm{sgn}(f_i(x) - x_i)$ for some $i \in \{1,\ldots,n\}$ satisfying $x_i \neq f_i(x)$, and $x'_j = x_j$ for all $j \neq i$. We call $S(f)$ the* asynchronous state transition graph of f.

To analyze state transition graphs we use, in addition to standard terminology from graph theory such as paths and cycles, the following concepts.

Definition 2.6 *An infinite path $(x^0,x^1,\ldots)$ in $S(f)$ is called* trajectory. *A nonempty set of states D is called* trap set *if every trajectory starting in D never leaves D. A trap set A is called* attractor *if for all $x^1, x^2 \in A$ there is a path from x^1 to x^2 in $S(f)$. Attractors of cardinality greater than one are called* cyclic *attractors. A state x is called* steady state*, if there exists an edge $x \to x$, i. e. if $f(x) = x$.*

It is easy to see that each trap set contains at least one attractor, and that attractors are the terminal strongly connected components of $S(f)$. They represent asymptotically stable behavior and often have clear biological meaning.

In Fig. 1 (c) we see the state transition graph for the network introduced in (a). The system has two cyclic attractors, namely $\{(1,1,0),(1,1,1)\}$ and $\{(2,1,0),(2,1,1)\}$.

As for the structural graph, we can define modules of the state transition graph as subgraphs, i. e. sets of states and corresponding state transitions representing fractions of the system's dynamics. However, it may also be of interest to only focus on the behavior of a subset of network components, which we can derive from the state transition graph by projection.

Definition 2.7 *Let $S' = (Y, E(S'))$ be a subgraph of $S(f)$. Let $\pi^I : X \to \prod_{i\in I} X_i$ be the projection on the components in the ordered set $I \subseteq \{1,\dots,n\}$. We define $\pi^I(S')$ as the graph with vertex set $\pi^I(Y)$ and edges $\pi^I(v^1) \to \pi^I(v^2)$ for $v^1, v^2 \in Y$ such that there exists an edge $v^1 \to v^2$ in $S(f)$, and $v^1 = v^2$ or $v_i^2 = v_i^1 + \operatorname{sgn}(f_i(v^1) - v_i^1)$ for some $i \in I$. We call $\pi^I(S')$* dynamical module of f.

Any subgraph of $S(f)$ is a dynamical module of f, whith I in the above definition chosen as the set $\{1,\dots,n\}$. Fig. 1 (e) shows a dynamical module of the function f given in the same figure. Here, we choose the subgraph S' consisting of the cyclic attractor $\{(1,1,0),(1,1,1)\}$ and the corresponding edges between the two attractor states. Then the dynamical module $\pi^{\{1,3\}}(S')$ of f is the graph in Fig. 1 (e).

Again, we do not incorporate any restrictions in the definition that ensure a significance of the modules, as e. g. in the approach in [6] where the authors focus on projected dynamics that are independent of the behavior of the rest of the system.

2.3 Network modules

As mentioned in the introduction, we are looking for subnetworks of f that are on the one hand easier to analyze than f itself, and on the other hand carry information of importance for understanding the original system. We define modules of the network f as follows.

Definition 2.8 *We call a function g a* network module of f, *if there exist $Y \subseteq X$ and an ordered set $I \subseteq \{1,\dots,n\}$ such that $g : \pi^I(Y) \to \pi^I(X)$ satisfies*

$$\pi^I \circ f|_Y = g \circ \pi^I|_Y,$$

where $|_Y$ denotes the restriction of a function to the set Y.
We call g autonomous, *if there exist integer intervals $Z_i = \{a_i, a_i+1, \dots, b_i\}$, $a_i \le b_i$, for all $i \in \{1,\dots,k\}$, $k = \operatorname{card} I$, such that $\pi^I(Y) = Z_1 \times \cdots \times Z_k$, and if $g(\pi^I(Y)) \subseteq \pi^I(Y)$.*

Let us again illustrate the notion using the example introduced in Fig. 1. For $Y = \{(1,1,0),(1,1,1)\}$, we have $f(Y) = \{(1,1,0),(1,1,1)\}$. If we set $I = \{1,3\}$ and $g : \{1\} \times \{0,1\} \to \{0,1,2\} \times \{0,1\}$ with $g((1,1)) = (1,0)$ and $g((1,0)) = (1,1)$, then $\pi^I(f(x)) = g(\pi^I(x))$ for $x \in Y$. Since $g(\pi^I(Y)) \subseteq \pi^I(Y)$ holds, g is an autonomous network module of f. Note that the set Y is a set on which the behavior of f can in some sense be characterized by the behavior of the components in I. If we add e. g. the state $(1,0,0)$ to the set Y in our example, then there is no function g satisfying $\pi^I \circ f|_Y = g \circ \pi^I|_Y$, since $\pi^I(1,0,0) = \pi^I(1,1,0)$, but $\pi^I(f(1,0,0)) = (1,0) \neq (1,1) = \pi^I(f(1,1,0))$. That is, we cannot distinguish the behavior of f in states $(1,0,0)$ and $(1,1,0)$ if we only have information on the components in I.

In general, network modules represent rather local aspects of the network in the sense that they describe the influences acting on a subset of components in a set of states. However, the information inherent in a network module does not necessarily suffice for determining dynamics beyond a single transition step. The second condition for autonomous network modules g allows to apply g iteratively

on states in $\pi^I(Y)$, while the first ensures that we can derive trajectories according to the asynchronous update rule in a projection of state space. Moreover, the first condition allows to apply Def. 2.2 to g. Thus, for autonomous network modules g we can determine an interaction graph $G(g)$ and a state transition graph $S(g)$. By abuse of notation we also denote $G(g)$ the graph derived from $G(g)$ by renaming the vertices $1,\ldots,k$ of $G(g)$ with the indices in I while preserving the order. This allows us to identify the interaction graph of g with a subgraph of $G(f)$.

Lemma 2.9 *Let g be an autonomous network module as introduced in Def. 2.8. Then $G(g)$ is a structural and $S(g)$ is a dynamical module of f.*

Proof As already mentioned, we associate each $i \in I$ with $l^i \in \{1,\ldots,k\}$ via an order-preserving mapping, and rename each vertex of $G(g)$ with indices in I according to this mapping. Obviously, the vertex set of $G(g)$ resp. $S(g)$ is a subset resp. a projection via π^I of a subset of the vertex sets of $G(f)$ resp. $S(f)$. Let $i \in I$, and choose $l^i \in \{1,\ldots,k\}$ as above, i. e., π^i maps the i-th component of a state $x \in X$ to the l^i-th component in $\pi^I(X)$. Then we have $f_i(y) = (\pi^I(f(y)))_{l^i} = g_{l^i}(\pi^I(y))$ for all $y \in Y$. Application of this equation to the conditions defining edges in Def. 2.2 and 2.5 easily renders that each edge in $G(g)$ is also an edge of $G(f)$, and that $S(g) = \pi^I(S')$, where S' denotes the subgraph of $S(f)$ with vertex set Y and edges $y^1 \to y^2$ of $S(f)$ with $y^1, y^2 \in Y$. □

For the network module g as defined as illustration for Def. 2.8 the state transition graph is shown in Fig. 1 (e). We rename the vertex set $\{1,2\}$ of $G(g)$ with $I = \{1,3\}$. The graph $G(g)$ then consists of the vertices α_1 and α_3 and a negative loop on α_3 as shown in Fig. 1(d).

In the following sections, we focus on developing a method to determine network modules useful in the analysis of f.

3 Symbolic steady states and frozen components

Often network components are involved in a number of specific tasks. Thus, although a network component may have a large range, only subsets of the range may be of interest when focusing on specific network behavior. To exploit this observation, we introduce the following notation. Here, we call the set $[a_i,b_i] := \{a_i, a_i+1,\ldots,b_i-1,b_i\} \subseteq X_i$ a discrete interval, if $a_i \leq b_i$, with $[a_i,a_i] := \{a_i\}$. In the following, we identify $\{a_i\}$ with a_i for all $a_i \in X_i$, $i \in \{1,\ldots,n\}$, and call a_i *regular value*. We will use intervals of cardinality greater than one instead of regular component values, if we do not have enough information to determine the exact component value. Following the terminology in [17, 16], we call intervals $[a_i,b_i]$ with $a_i < b_i$ *symbolic values*.

We now need to integrate symbolic values in the dynamical analysis. Here, we generalize ideas from [17, 16].

Definition 3.1 *For every $i \in \{1,\ldots,n\}$ let $X_i^\square$ denote the set $\{[a_i,b_i] \subseteq X_i \mid a_i \leq b_i\}$ of discrete intervals in the range X_i. Set $X^\square = X_1^\square \times \cdots \times X_n^\square$. We call elements in X* regular*, elements in $X^\square \setminus X$* symbolic states*. By $J(M)$ we denote the set of all symbolic valued components of M for $M \in X^\square$. Define*

$$F : X^\square \to X^\square,\ M \mapsto (F_1(M),\ldots,F_n(M)) \text{ with } F_i(M) = [\min_{x\in M} f_i(x), \max_{x\in M} f_i(x)] \text{ for all } i \in \{1,\ldots,n\}.$$

We call a state $M \in X^\square \setminus X$ satisfying $F(M) = M$ symbolic steady state*.*

The elements of $X^{\square}$ are subsets of X. The functions f and F coincide on the set X of regular states which we identify with the elements of $X^{\square}$ of cardinality one. In general, if a component function value $f_i(M)$ is regular, this means there is enough information inherent in M to exactly specify its value, while a symbolic value represents the fact that we have not enough information to do so. However, it may be possible to at least derive some constraint for the function value represented by the interval boundaries. For our running example given in Fig. 1 the state $([1,2],1,[0,1]) = ([1,2],1,X_3) = F(([1,2],1,X_3))$ is a symbolic steady state, where we can determine $F_2(([1,2],1,[0,1])) = 1$ exactly, obtain the constraint that the first component cannot have value 0, but have no information on the third component.

We are particularly interested in regular components of a symbolic state M that remain fixed on all trajectories starting in M. Keeping in mind that we consider the asynchronous update strategy, we can find a superset of the set of states reachable from M by the following procedure. We define $\tilde{M}^0 := M$ and $\tilde{M}_j^k := [\min(\tilde{M}_j^{k-1} \cup F_j(\tilde{M}^{k-1})), \max(\tilde{M}_j^{k-1} \cup F_j(\tilde{M}^{k-1}))]$ for all $k \in \mathbb{N}$. Since the boundaries of the intervals M_j^k decrease resp. increase monotonously and are bounded by 0 resp. the maximal activity level p_j, the sequence $(\tilde{M}^k)_{k\in\mathbb{N}_0}$ converges to a symbolic state $\widetilde{M}$ representing a superset of the set of from M in $S(f)$ reachable states. In particular, no trajectory starting in $\widetilde{M}$ can leave $\widetilde{M}$. We call $\widetilde{M}$ *extended forward orbit of* M. The next definition is in reference to the notion of frozen cores in random Boolean networks introduced by S. Kaufman [8].

Definition 3.2 *Let $i \in \{1,\dots,n\}$. If M is a symbolic state with regular component M_i such that $\widetilde{M}_i = M_i$ for the extended forward orbit $\widetilde{M}$ of M, then we say that the i-th network component is a* frozen component *of M, or a component* frozen to value M_i. *The set I of all frozen components of a symbolic state M is called* frozen core *of M, and is denoted by (I,M).*

If a component j of a symbolic state M has symbolic value X_j, then of course the j-th component of the extended forward orbit is also X_j. For the example network in Fig. 1 the symbolic state $M = (X_1,1,X_3)$ coincides with its extended forward orbit. Thus, the frozen core of M is given by $(\{2\},(X_1,1,X_3))$.

Clearly, the frozen core of a symbolic steady state coincides with its set of regular components. Moreover, we can use the frozen core of a symbolic state to obtain a symbolic steady state, as the next statement shows.

Theorem 3.3 *Let (I,M') be the frozen core of a symbolic state $M' \in X^{\square}$. Set $M^0 := \widetilde{M'}$ and $M^k := F(M^{k-1})$ for all $k \in \mathbb{N}$. Then $(M^k)_{k\in\mathbb{N}}$ converges to a regular or a symbolic steady state M. We call M the (symbolic) steady state derived from (I,M').*

Proof If the sequence converges to a limit M, clearly $F(M) = M$ follows from the definition of the sequence. The state M is a regular or a symbolic steady state depending on the cardinality of M. We show convergence of $(M^k)_{k\in\mathbb{N}}$ by proving via induction that the sequence is decreasing monotonously with respect to the subset relation. That is, we show $M^{l+1} \subseteq M^l$ for all $l \in \mathbb{N}$.

For $i \in I$, we have $M_i^1 = M_i^0$ by the definition of frozen components. If $i \in J(M^0) = \{1,\dots,n\} \setminus I$, we have $M_i^1 \subseteq M_i^0$ by the definition of the extended forward orbit.

Now, let $l \in \mathbb{N}$ and assume $M^k \subseteq M^{k-1}$ for all $k \leq l$. Recall that $M_i^{l+1} = F(M^l)_i = [\min_{x\in M^l} f_i(x), \max_{x\in M^l} f_i(x)]$ and $M_i^l = F(M^{l-1})_i = [\min_{x\in M^{l-1}} f_i(x), \max_{x\in M^{l-1}} f_i(x)]$ for all $i \in \{1,\dots,n\}$. Since $M^l \subseteq M^{l-1}$, we have $\min_{x\in M^{l-1}} f_i(x) \leq \min_{x\in M^l} f_i(x) \leq \max_{x\in M^l} f_i(x) \leq \max_{x\in M^{l-1}} f_i(x)$, and thus $M_i^{l+1} \subseteq M_i^l$ for all $i \in \{1,\dots,n\}$. □

As mentioned above, for our running example $(\{2\},(X_1,1,X_3))$ is the frozen core of the state $(X_1,1,X_3)$, where $X_1 = \{0,1,2\}$ and $X_3 = \{0,1\}$. Since the state coincides with its extended forward orbit, we start

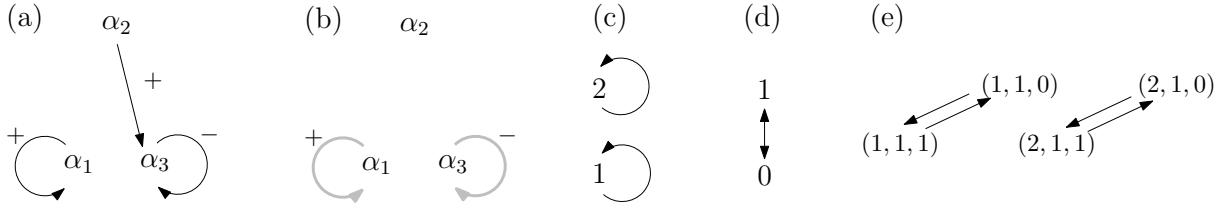


Figure 2: Consider f as given in Fig. 1. In (a) the local interaction graph $G((1,1,0))$ of f. In (b) the graph $G(f|_M)$ for the symbolic steady state $M = ([1,2],1,[1,0])$. Here, heavier gray edges indicate the two components of $G^\theta(M)$. In (c), (d) and (e) the state transition graphs $S(f^{Z_1})$, $S(f^{Z_2})$ and S^M, respectively.

the sequence with $M^0 = (X_1,1,X_3) = ([0,2],1,[0,1])$, and $M^1 = ([1,2],1,[0,1]) = M$ is the symbolic steady state derived from $(\{2\},(X_1,1,X_3))$.

4 Network analysis using modules

In the following we want to determine network modules, such that the results of the module analysis can be directly used to obtain information on the behavior of the original system. Clearly, identification of such modules is generally only possible if we exploit at least some coarse knowledge of structural as well as dynamical characteristics of the original system. It turns out that the information inherent in a symbolic steady state M is sufficient to determine network modules. We proceed by first associating a dynamical module with M, then derive a structural module, and finally define a network module suitable for utilization in network analysis.

Let us start by analyzing dynamical characteristics associated with a symbolic steady state M. If $x \in M$ is a regular state, then $f(x) \in M$ by definition of F. More precisely, $x_i + \mathrm{sgn}(f_i(x) - x_i) \in M_i$, since the interval bounded by x_i and $f_i(x)$ is a subset of M_i. Thus, every trajectory starting in M remains in M. We formulate this fact in the following statement.

Proposition 4.1 *If M is a symbolic steady state, then the set of regular states represented by M is a trap set.*

By definition, the subgraph of $S(f)$ with vertex set M and all edges in $S(f)$ with both end-vertices in M is a dynamical module. The result above shows that this module is of significance for the dynamical analysis of f, since a trap set always contains at least one attractor.

In order to associate a structural module with M, we have to recognize some important properties attached to the regular components of M. While symbolic components may be dynamically active in the trap set M, i. e., the components change their values along at least some trajectories in M, the regular components remain fixed regardless of the behavior of the symbolic components in M. This means that the network components with symbolic values, i. e., components in $J(M)$, do not influence the behavior of the network components with regular values in the trap set M. In turn, the influence of the regular on the symbolic components remains the same for all states in M. This motivates the following definition describing the structural modules associated with M.

Definition 4.2 *Let M be a symbolic steady state. By $G^\theta(M)$ we denote the (multi-)graph with vertex set $V^\theta(M) = J(M)$ and edge set $E^\theta(M) = \{(i,j,\varepsilon) \in E^M \mid i,j \in J(M)\}$, where E^M denotes the edge set of*

$G(f|_M)$. *We call a graph* $Z = (V^Z, E^Z)$ component *of* $G^\theta(M)$*, if the undirected graph derived from* Z *is a maximal connected subgraph of the undirected graph derived from* $G^\theta(M)$.

Note that we use the global interaction graph $G(f|_M)$ instead of the local interaction graph $G(M)$ in the definition. The difference is that we only consider edges derived from component value changes in M instead of in X (compare Def. 2.2 and 2.3), and thus capture all interactions functional in M. In particular, there are no edges originating in frozen components of M, since it is not possible to vary their value without leaving M.

To illustrate the above notions let us again consider the example introduced in Fig. 1. We have seen that the state $M = ([1,2],1,[1,0])$ is a symbolic steady state. In Fig. 1 (c) we can easily see that the set of regular states $x \in M$ is a trap set which contains both attractors of the system. The global interaction graph $G(f|_M)$ is shown in Fig. 2 (b). We obtain $G^\theta(M)$ simply by eliminating the vertex α_2. The two components of $G^\theta(M)$ are the loops originating in α_1 resp. α_3.

The components of $G^\theta(M)$ are the structural modules we associate with M. In preparation for our definition of a network module derived from a structural module we need to verify that network components belonging to different components of $G^\theta(M)$ do not influence each others behavior in M. This property is captured in the following lemma, which has already been proved under slightly different conditions in [17, 16].

Lemma 4.3 *Let* M *be a symbolic steady state, and let* $Z_1, \ldots, Z_k$ *be the components of* $G^\theta(M)$*. Consider a union* Z *of arbitrary components* Z_j*. Let* $x, y \in M$ *such that* $x_i = y_i$ *for all* $i \notin Z$*. Then* $f_i(x) = f_i(y)$ *for all* $i \notin Z$*. In particular, for* $M' \in X^\square$ *such that* $M'_i = M_i$ *for all* $i \notin Z$ *and* $M'_i \subseteq M_i$ *for* $i \in Z$*, we have* $F_i(M') = F_i(M) = M_i = M'_i$ *for all* $i \notin Z$.

Proof For $i \notin J(M)$, i. e., for a frozen component i of M, we have $f_i(x) = f_i(y) = F_i(M)$ since $x, y \in M$.

Let $i \in J(M) \setminus Z$, and assume $f_i(x) \neq f_i(y)$. We know that if $x_j \neq y_j$ then $j \in Z$. Since $x, y \in M$, we can define a sequence $(x = x^1, x^2, \ldots, x^m = y)$ in M such that x^l and x^{l+1} differ in one component only, which is in Z, and the corresponding component values differ by absolute value one. Since $f_i(x) \neq f_i(y)$, it follows that $f_i(x^l) \neq f_i(x^{l+1})$ for some $l < m$. According to Def. 2.2 there exists an edge in $G(f|_M)$ from some component in Z to i, which is a contradiction. □

The lemma shows that the frozen core of M constitutes a boundary between the components of $G^\theta(M)$ that enables us to analyze their behavior in isolation from each other. To do so, we now derive a network module from a component Z of $G^\theta(M)$ by defining a function f^Z.

Lemma 4.4 *Let* M *be a symbolic steady state, and let* $Z = (V^Z, E^Z)$ *be a component of* $G^\theta(M)$*. Set* $k^Z = \operatorname{card} V^Z$*. Let* ι^Z *be an order preserving bijection from* $\{1, \ldots, k\}$ *to* V^Z*. Set* $X^Z = M_{\iota^Z(1)} \times \cdots \times M_{\iota^Z(k)}$*. We define* $f^Z : X^Z \to X^Z$*,* $f^Z = \pi^Z \circ F \circ \rho^Z$*, where* $\rho^Z : X^Z \to X^\square$ *with* $\rho^Z_i(z) = M_i$ *for* $i \notin Z$ *and* $\rho^Z_i(z) = z_{\iota^Z(i)}$ *for* $i \in Z$*, and* $\pi^Z : X^\square \to X^Z$ *is the projection on the components of* Z*.*
Then f^Z *is an autonomous network module, and is called the* network module of f derived from Z.

Proof The function f^Z maps regular states to regular states, since Z is disjoint from $J(M) \setminus Z$ in $G^\theta(M)$ and thus $F_i(\rho^Z(z)) \in X_i$ for $z \in X^Z$, $i \in Z$, according to Lemma 4.3 and the definition of F. We now show $\pi^Z \circ f|_M = f^Z \circ \pi^Z|_M$. Let $j \in \{1, \ldots, k\}$ and $x \in M$. To simplify notation we drop the superscript Z from π^Z, ρ^Z and ι^Z. We have $F_{\iota(j)}(\rho(\pi(x))) = F_{\iota(j)}(x) = f_{\iota(j)}(x)$ according to Lemma 4.3 and the definition of ρ and F. Thus, $(\pi^Z \circ f|_M(x))_j = f_{\iota(j)}(x) = F_{\iota(j)}(x) = F_{\iota(j)}(\rho(\pi(x))) = (\pi \circ F \circ \rho(\pi(x)))_j = f^Z_j(\pi(x))$. Furthermore, since M is a symbolic steady state $X^Z = \pi(M)$ satisfies the conditions regarding the domain

of an autonomous network module given in Def. 2.8. □

It is easy to see that the global interaction graph $G(f^Z)$ is isomorphic to Z, so the structural module derived from M matches the one derived from f^Z. The dynamical modules derived from all functions f^Z, Z component of $G^\theta(M)$, i. e., the state transition graphs $S(f^Z)$, constitute a breakdown of the coarse dynamical module, i. e., the trap set M, which we used to determine first the structural and then the network modules. We end this section by showing that these finer dynamical modules are building blocks of the dynamics of f. More specific, we show that we can compose the state transition graph of f from the state transition graphs of the network modules f^Z and the frozen components of M. Again, in the following we generalize results from [17]. First, we define the composition of the graphs $S(f^Z)$.

Definition 4.5 *Let $M \in X^\square$ be a symbolic steady state, and let $Z_1,\ldots,Z_m$ be the components of $G^\theta(M)$. We then denote by S^M the graph with vertex set M and edge set E^M. An edge $x^1 \to x^2$ belongs to the edge set iff*

$$x^1 = x^2, \text{ and } \quad \pi^{Z_j}(x^1) \to \pi^{Z_j}(x^2) \quad \text{belongs to } S(f^{Z_j}) \text{ for all } j \in \{1,\ldots,m\},$$

or if there exists $j \in \{1,\ldots,m\}$ such that

$$\pi^{Z_j}(x^1) \to \pi^{Z_j}(x^2) \text{ is an edge in } S(f^{Z_j}) \text{ and } x^1_i = x^2_i \text{ for all } i \notin V^{Z_j}.$$

We call S^M the product state transition graph corresponding to M.

The next theorem confirms that the method of composing the state transition graphs of the network modules renders the subgraph of $S(f)$ derived from the state set M.

Theorem 4.6 *Let $M \in X^\square$ be a symbolic steady state, and let $Z_1,\ldots,Z_m$ be the components of $G^\theta(M)$. Let $S(f)|_M$ denote the subgraph of $S(f)$ with vertex set M and all edges in $S(f)$ with both end-vertices in M. Then $S(f)|_M = S^M$.*

Proof Recall that M is a trap set, and that for $x,x' \in M$ there is an edge $x \to x'$ in $S(f)$ if and only if $x' = f(x) = x$ or $x'_i = x_i + \operatorname{sgn}(f_i(x) - x_i)$ for some $i \in \{1,\ldots,n\}$ satisfying $x_i \neq f_i(x)$, and $x'_j = x_j$ for all $j \neq i$.

First, we note that $f_i(x) = x_i = M_i$ for all $x \in M$ and $i \notin J(M)$. Furthermore, for $i \in J(M)$ there exists $k^i \in \{1,\ldots,m\}$ such that $i \in V^{Z_{k^i}}$, and there exists l^i such that $\iota^{Z_{k^i}}(l^i) = i$, with $\iota^{Z_{k^i}}$ being the bijection introduced in Def. 4.4. As seen in the proof of Lemma 2.9, we have $f_i(x) = f^{Z_{k^i}}_{l^i}(\pi^{Z_{k^i}}(x))$. Therefore, $x \in M$ is a fixed point of f iff $\pi^{Z_j}(x)$ is a fixed point of f^{Z_j} for all $j \in \{1,\ldots,m\}$, and $x_i + \operatorname{sgn}(f_i(x) - x_i) = (\pi^{Z_{k^i}}(x))_{l^i} + \operatorname{sgn}(f^{Z_{k^i}}_{l^i}(\pi^{Z_{k^i}}(x)) - (\pi^{Z_{k^i}}(x))_{l^i})$. Thus, we can construct each edge in $S(f)|_M$ from an edge in some $S(f^{Z_j})$ and vice versa. It follows from Def. 4.5 that $S(f)|_M = S^M$. □

The reasoning in the proof of Theorem 4.6 leads immediately to the following statement.

Corollary 4.7 *Let M and $Z_1,\ldots,Z_m$ be as in Theorem 4.6. For all $i \in \{1,\ldots,m\}$ let A_i be an attractor in $S(f^{Z_i})$. Then $A := \{a \in M \mid \forall j \in \{1,\ldots,m\} : \pi^{Z_j}(a) \in A_j\}$ is an attractor in $S(f)$. Moreover, every attractor in $S(f)|_M$ can be represented in this manner as product of attractors in $S(f_{Z_j})$, $j \in \{1,\ldots,m\}$, and component values M_i for $i \notin J(M)$.*

We illustrate the results on our running example from Fig. 1. The graph $G^\theta(M)$ for $M = ([1,2],1,[0,1])$ has two components, Z_1 consisting of a positive loop on α_1, and Z_2 being a negative loop on α_3, as can be seen in Fig. 2 (b). We define $f^{Z_1} : [1,2] \to [1,2]$ as in Def. 4.4 with $\pi^{Z_1}((M'_1,M'_2,M'_3)) = M'_1$ for all

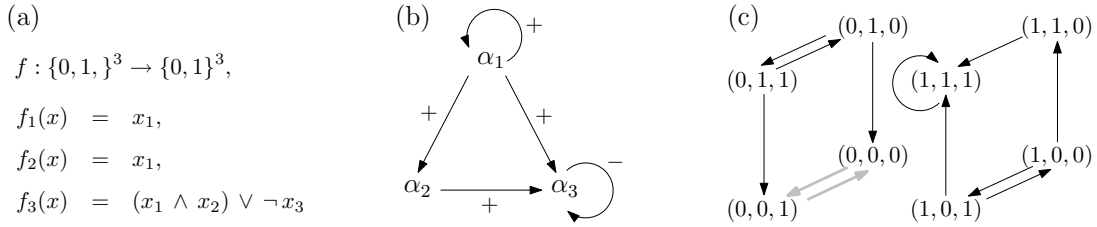


Figure 3: A network with input layer in (a), its global interaction graph in (b), and its state transition graph in (c).

$M' \subseteq M$ and $\rho^{Z_1}(z) = (z, 1, [0,1])$ for all $z \in [1,2]$. This reduces the function f_1 given in Fig. 1 (a) to $f^{Z_1}(z) = z$. The state transition graph $S(f^{Z_1})$ is given in Fig. 2 (c) and consists of two steady states. Analogously, we obtain $S(f^{Z_2})$, which comprises a single attractor of cardinality 2 as shown in Fig. 2 (d). Applying Def. 4.5 we obtain the graph S^M which is shown in Fig. 2 (e). Comparison with the state transition graph $S(f)$ given in Fig. 1 (c) illustrates Theorem 4.6 and its corollary.

5 Networks with input layer

In this section we introduce a class of networks, for which we can easily find a set of symbolic steady states such that all attractors of the original system can be constructed from the attractors of the network modules derived from the symbolic steady states. We extend results obtained in [17].

Definition 5.1 *We call f a* network with input layer*, if there exists $i \in \{1,\ldots,n\}$ such that $f_i = \mathrm{id}_{X_i}$. A vertex satisfying this condition is called* input vertex.

For every input vertex α_i, we have no incoming edges except a positive loop in the global interaction graph. However, this structural criterion is not sufficient for identifying an input vertex. Networks with input layer are well-suited for modeling, for example, signal transduction networks, where receptors may be modeled as input vertices.

In the following we assume that f is a network with input layer. Without loss of generality we assume that $\alpha_1,\ldots,\alpha_k$, $k \in \{1,\ldots,n\}$, are the input vertices of f.

We can immediately note one important property of f. The frozen core of a symbolic state $M' \in X^{\square}$ with $M'_i \in X_i$ for $i \in \{1,\ldots,k\}$ and $M'_i = X_i$ for $i > k$ is given by $(\{1,\ldots,k\}, M')$, since M' is its own extended orbit. In other words, we can easily derive a (symbolic) steady state from each combination of values for the input vertices using Theorem 3.3. The resulting set of symbolic and possibly regular steady states is sufficient to determine all attractors of f.

Theorem 5.2 *Let A be an attractor of f. Then there exist input values $x_i \in X_i$, $i \in \{1,\ldots,k\}$ such that either $A = M$ is a regular steady state, or we can construct A from M as shown in Cor 4.7, where M is the fixed point derived from $(\{1,\ldots,k\}, M')$, $M' \in X^{\square}$ with $M'_i = x_i$ for $i \in \{1,\ldots,k\}$ and $M'_i = X_i$ for $i > k$.*

Proof Since $\alpha_1 \ldots \alpha_k$ are input vertices, we have $a_i = a'_i$ for all $i \in \{1,\ldots,k\}$, and we set $M'_i = a_i$ for the input vertices. Next, we show that for every state $x \in X$ with $x_i = M'_i$ for all $i \in \{1,\ldots,k\}$, there exists a trajectory leading to M.

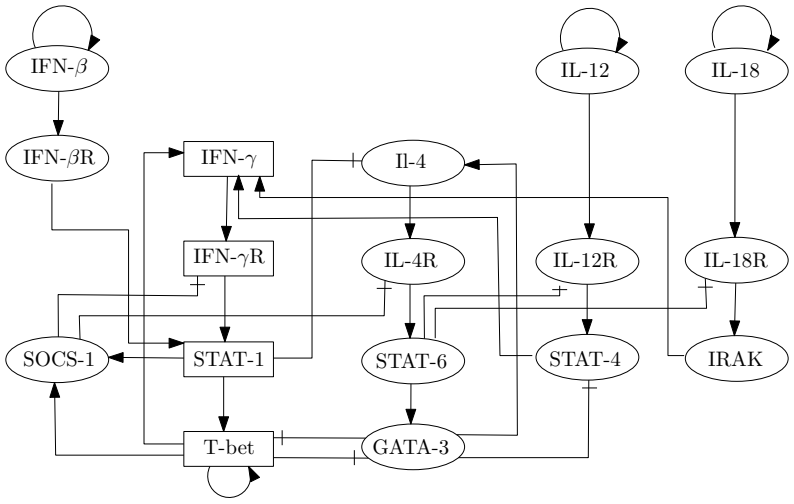


Figure 4: Global interaction graph of the Th cell differentiation network introduced in [10]. Arrows represent activation, crossed lines represent inhibition.

Let $(M^l)_{l\in\mathbb{N}_0}$ with $M^0 := \widetilde{M'}$ and $M^l := F(M^{l-1})$ for all $l \in \mathbb{N}$ be the sequence converging to M introduced in Theorem 3.3. Recall that $M^l \subseteq M^{l-1}$ for all $l \in \mathbb{N}$. Let $x^0 \in M' \setminus M$. Then there exists $l \in \mathbb{N}_0$ such that $x^0 \in M^l \setminus M^{l+1}$. Then, according to the definition of F and since $(M^l)_{l\in\mathbb{N}_0}$ is decreasing with respect to the subset relation, either all successors of x^0 in $S(f)$ are in $M^l \setminus M^{l+1}$ or there exists a successor in M^{l+1}. In the latter case, we label that successor x^1. Otherwise, we repeat the procedure for all successor of x^0, check again and if necessary repeat again. Since all images of all states in M^l lie in M^{l+1} and since the state space is finite, we eventually find a state $x^1 \in M^{l+1}$ such that there exists a path from x^0 to x^1 with all states of the path except x^1 lying in $M^l \setminus M^{l+1}$. Since $(M^l)_{l\in\mathbb{N}_0}$ is converging to M we can thus construct a path from each state in M' to M.

It follows that there is no trap set, and thus no attractor, in $M' \setminus M$. If M is a regular steady state, then $M = A$ is the only attractor in M'. Otherwise A is a composition of attractors of network modules derived from M and the frozen components of M as shown in Cor 4.7. □

We illustrate the results on the simple Boolean network given in Fig. 3 which has one input vertex, namely α_1. We start calculating the two (symbolic) steady states from the input values $x_1 = 0$ and $x_1 = 1$. In the first case, we get $F((0,[0,1],[0,1])) = (0,0,[0,1])$ and $F((0,0,[0,1])) = (0,0,[0,1])$, i. e., $(0,0,[0,1])$ is a symbolic steady state, and the associated structural module is a negative loop originating in α_3. When looking at the corresponding network module, we obtain the function $f^Z : [0,1] \to [0,1]$, $f^Z(z) = \neg z$, and the derived attractor for the original network is the set $\{(0,0,0),(0,0,1)\}$. For the input value $x_1 = 1$ we get the sequence $((1,[0,1],[0,1]),(1,1,[0,1]),(1,1,1),(1,1,1),\dots)$, so the procedure renders a regular steady state of the system. As can be seen in Fig. 3, the regular steady state and the attractor derived from $(0,0,[0,1])$ are the only attractors of the system.

6 Analyzing Th cell differentiation

T helper cells, short Th cells, are important players in the vertebrate immune system. They can be sub-classified in Th1 and Th2 cells, which are involved in different immune responses. Both originate

IFN-β	$X_1 = \{0,1\}$	$f_1(x) = x_1$
IL-12	$X_2 = \{0,1\}$	$f_2(x) = x_2$
IL-18	$X_3 = \{0,1\}$	$f_3(x) = x_3$
IFN-βR	$X_4 = \{0,1\}$	$f_4(x) = x_1$
IFN-γ	$X_5 = \{0,1,2\}$	$f_5(x) = 1$ if $(x_{16} = 1 \wedge \neg(x_{14} = 1 \wedge x_{15} = 1)) \vee (x_{14} = 1 \wedge x_{15} = x_{16} = 0)$, $f_5(x) = 2$ if $x_{16} = 2 \vee (x_{14} = 1 \wedge x_{15} = 1)$, and $f_5(x) = 0$ otherwise
IL-4R	$X_6 = \{0,1\}$	$f_6(x) = 1$ if $x_{12} = 0 \wedge x_{17} = 1$, and $f_6(x) = 0$ otherwise
IFN-γR	$X_7 = \{0,1,2\}$	$f_7(x) = 1$ if $x_5 = 1 \vee (x_5 = 2 \wedge x_{11} = 1)$, $f_7(x) = 2$ if $x_5 = 2 \wedge x_{11} = 0$, and $f_7(x) = 0$ otherwise
IL-4R	$X_8 = \{0,1\}$	$f_8(x) = x_6 \wedge \neg x_{11}$
IL-12R	$X_9 = \{0,1\}$	$f_9(x) = x_2 \wedge \neg x_{13}$
IL-18R	$X_{10} = \{0,1\}$	$f_{10}(x) = x_3 \wedge \neg x_{13}$
SOCS-1	$X_{11} = \{0,1\}$	$f_{11}(x) = 1$ if $x_{12} \geq 1 \vee x_{16} \geq 1$, and $f_{11}(x) = 0$ otherwise
STAT-1	$X_{12} = \{0,1,2\}$	$f_{12}(x) = 1$ if $(x_4 = 1 \wedge x_7 = 0) \vee x_7 = 1$, $f_{12}(x) = 2$ if $x_7 = 2$, and $f_{12}(x) = 0$ otherwise
STAT-6	$X_{13} = \{0,1\}$	$f_{13}(x) = x_8$
STAT-4	$X_{14} = \{0,1\}$	$f_4(x) = x_9 \wedge \neg x_{17}$
IRAK	$X_{15} = \{0,1\}$	$f_{15}(x) = x_{10}$
T-bet	$X_{16} = \{0,1,2\}$	$f_{16}(x) = 1$ if $(x_{17} = 0 \wedge ((x_{12} = 1 \wedge x_{16} \leq 1) \vee (x_{12} \leq 1 \wedge x_{16} = 1)))$ $\vee (x_{17} = 1 \wedge x_{16} = 1 \wedge x_{12} = 1)$, $f_{16} = 2$ if $(x_{17} = 0 \wedge (x_{12} = 2 \vee x_{16} = 2)) \vee (x_{17} = 1 \wedge x_{12} = 1 \wedge x_{16} = 2)$, $f_{16}(x) = 0$ otherwise
GATA-3	$X_{17} = \{0,1\}$	$f_{17}(x) = 1$ if $x_{13} = 1 \wedge x_{16} = 0$, and $f_{17}(x) = 0$ otherwise

Table 1: Coordinate functions and ranges for the components of the Th cell network.

from a common precursor, promote their own differentiation and inhibit proliferation of each other. In [10] L. Mendoza proposes a model for a control network of Th cell differentiation consisting of 17 components, 13 of which are represented by Boolean variables while the remaining 4 components have three activity levels. The logical rules governing the behavior of the system are given in Table 1 and the global interaction graph can be seen in Fig. 4. Note that the model depicted in Fig. 4 and Table 1 differs slightly from the model introduced in [10], namely we altered the logical functions associated with the vertices IFN-β, IL-12 and Il-18. In Mendoza's model, the three components are modeled with constant functions with value 0, representing the wild type in some sense (see [10]). The constant values are changed when considering specific artificial environmental conditions. We model these vertices as input vertices and consider for the wild type the situation where all input values are zero. Clearly, the attractors of both models coincide. Modeling IFN-β, IL-12 and IL-18 as input vertices also makes sense from a biological point of view since all three vertices represent substances not reproduced by Th cells. If we want to mimic experimental conditions where cells are cultured in media saturated with one or more of these substances, we can easily do so by focussing on the part of state space where one or more of the input vertices have value one. Note that we have no further input vertices in our model.

For the wild type, i. e. the situation where all input values are set to zero, Mendoza identifies four attractors all of which are fixed points of the function f given in Table 1. Each one has a clear biological interpretation [10]. We now want to apply our analysis technique using symbolic steady states to the wild type.

We fix the values of the input vertices to zero and as a first step determine the corresponding symbolic steady state, that is, the symbolic steady state derived from the frozen component set $(\{1,2,3\}, x_1 = x_2 = x_3 = 0)$. Iterating the state $M^0 := (0,0,0,[0,1],[0,2],[0,1],[0,2],[0,1],[0,1],[0,1],[0,1],[0,2],[0,1],[0,1],[0,1],[0,2],[0,1])$ we get

$$\begin{aligned}
M^1 &:= f(M^0) = (0,0,0,0,[0,2],[0,1],[0,2],[0,1],0,0,[0,1],[0,2],[0,1],[0,1],[0,1],[0,2],[0,1]),\\
M^2 &:= f(M^1) = (0,0,0,0,[0,2],[0,1],[0,2],[0,1],0,0,[0,1],[0,2],[0,1],0,0,[0,2],[0,1]),\\
f(M^2) &= M^2.
\end{aligned}$$

We obtain a symbolic steady state with 8 regular components, and no further constraints on the remaining components. The local interaction graph $G^\theta(M)$ is shown in Fig. 5 (a). Analysis of the corresponding subnetwork renders four fixed points, namely $(0,0,0,0,0,0,0,0,0,0)$, $(1,0,1,0,1,1,0,1,0)$, $(2,0,1,0,1,1,0,2,0)$, $(0,1,0,1,0,0,1,0,1) \in X_5 \times X_6 \times X_7 \times X_8 \times X_{11} \times X_{12} \times X_{13} \times X_{16} \times X_{17}$. The steady states of the original network derived from these fixed points match the four steady states found in [10]. The state space of the original model consists of 663552 states. Fixing the input values still leaves us with 82944 states to consider. The state space of the structural module associated with the symbolic steady state M^2 contains only 2592 states.

Not all of the combinations of input values render a significant simplification of the network analysis. In the worst case, for example if we choose input values $x_1 = 0$, $x_2 = x_3 = 1$, we can only derive the value for x_4 but no further constraints on structure and behavior of the system. This in itself is of course an interesting observation from a biological point of view, since in that case we can deduce that cross-regulation plays an important role at an early stage of signal transduction.

On the other hand, some combinations of input values lead to very small network modules. Let us as a last example consider the input values $x = 1$, $x_2 = x_3 = 0$, representing an overabundance of IFN-β. Starting with $M^0 := (1,0,0,[0,1],[0,2],[0,1],[0,2],[0,1],[0,1],[0,1],[0,1],[0,2],[0,1],[0,1],[0,1],[0,2],[0,1])$ we get

$$\begin{aligned}
M^1 &:= f(M^0) = (1,0,0,1,[0,2],[0,1],[0,2],[0,1],0,0,[0,1],[0,2],[0,1],[0,1],[0,1],[0,2],[0,1]),\\
M^2 &:= f(M^1) = (1,0,0,1,[0,2],[0,1],[0,2],[0,1],0,0,[0,1],[1,2],[0,1],0,0,[0,2],[0,1]),\\
M^3 &:= f(M^2) = (1,0,0,1,[0,2],0,[0,2],[0,1],0,0,1,[1,2],[0,1],0,0,[0,2],[0,1]),\\
M^4 &:= f(M^3) = (1,0,0,1,[0,2],0,[0,1],0,0,0,1,[1,2],[0,1],0,0,[0,2],[0,1]),\\
M^5 &:= f(M^4) = (1,0,0,1,[0,2],0,[0,1],0,0,0,1,1,0,0,0,[0,2],[0,1]),\\
M^6 &:= f(M^5) = (1,0,0,1,[0,2],0,[0,1],0,0,0,1,1,0,0,0,[0,2],0),\\
M^7 &:= f(M^6) = (1,0,0,1,[0,2],0,[0,1],0,0,0,1,1,0,0,0,[1,2],0),\\
M^8 &:= f(M^7) = (1,0,0,1,[1,2],0,[0,1],0,0,0,1,1,0,0,0,[1,2],0),\\
M^9 &:= f(M^8) = (1,0,0,1,[1,2],0,1,0,0,0,1,1,0,0,0,[1,2],0),\ f(M^9) = M^9.
\end{aligned}$$

We obtain a two-component module consisting of a positive loop on T-bet and an activating edge from T-bet to IFN-γ which can be seen in Fig. 5 (b). Both components of the module originally have three activity levels, but are both constrained to levels 1 and 2 in the module dynamics. Thus, we only have to

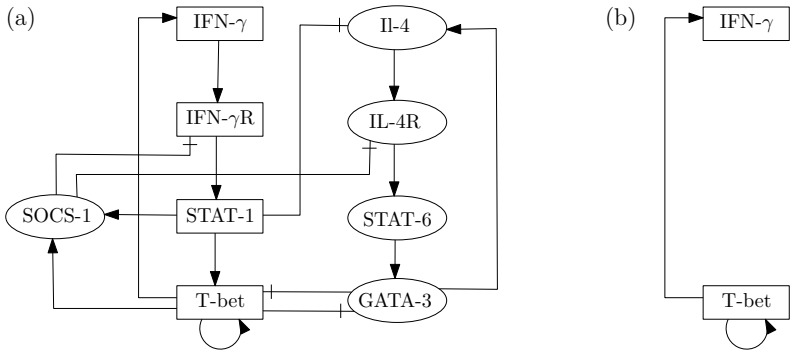


Figure 5: Subnetworks of the Th cell differentiation network associated with the symbolic fixed points derived from the input values $x_1 = x_2 = x_3 = 0$ in (a) and $x_1 = 1, x_2 = x_3 = 0$ in (b).

analyze a state space of cardinality four instead of a state space consisting of 82944 states. The module has two steady states, namely $(x_5, x_{16}) = (1,1)$ and $(x_5, x_{16}) = (2,2)$, which translate to two steady states in the original network. Again, this is in agreement with the results in [10] (supplementary material).

Application of our analysis method to this model thus offers two advantages. First, the complexity of the analysis of the dynamics is reduced, since we only have to focus on the smaller network modules. Furthermore, identification of the modules themselves is of interest, since they represent the part of the system responsible for the decision of the system's fate, i. e., which attractor is reached. An interesting next step would then be to check whether the mathematically derived network modules coincide with subsystems of known biological importance.

7 Conclusion

In this paper we introduced a method to analyze discrete regulatory networks utilizing suitable network modules. We used the notion of symbolic steady state, which allows us to determine such network modules using coarse dynamical and subsequently structural modules derived from the symbolic steady state. Lastly, we can associate network modules with the structural modules exploiting the properties of the frozen core of the symbolic steady state. We then can construct the dynamics of the original network, and in particular its attractors, in a subset of state space explicitly from the state transition graphs of the network modules. This paper not only gives a rigorous definition of different aspects of modularity but notably extends results in [17, 16]. In particular, the detailed analysis of the Th cell network becomes possible because of the refined notion of symbolic steady state.

A variety of aspects provide possibilities for fruitful future work. Firstly, we want to focus on further options for easily computing symbolic steady states. Here, we introduced a method suited for networks with input-layer, but the resulting symbolic steady states might not be minimal with respect to the subset relation on symbolic steady states, and thus the resulting network modules can possibly be further refined. We anticipate results when we focus on certain classes of functions f describing networks, in particular (nested) canalyzing functions [8, 7]. A different direction of interest is to use the network modules not only for obtaining the system's dynamics but also for a refined stability analysis. Perturbations resulting in changes in the dynamics of f might not be noticeable in every network module (see [6]). Such considerations are of similar interest when using the synchronous update strategy, which is also often utilized in discrete modeling, so a translation of our results to synchronous update networks seems worthwhile.

Furthermore, we need to compare our results to other well-established modularization techniques that aim at reducing the analysis complexity such as the modular response analysis of biochemical networks [9, 2]. Although the underlying modeling approaches are very different, some ideas may be transferable or render complementing results. Lastly, we plan to apply our methods to further biological examples. Here, a comparison of the network modules and their associated structural and dynamical modules with subsystems of known biological importance could lead to interesting insights.

References

[1] U. Alon (2007): *Network motifs: theory and experimental approaches*. Nature Reviews Genetics 8, pp. 450–461.

[2] F. J. Bruggeman, H. V. Westerhoff, J. B. Hoek & B. N. Kholodenko (2002): *Modular Response Analysis of Cellular Regulatory Networks*. J. Theor. Biol. 218(4), pp. 507 – 520.

[3] C. Christensen & R. Albert (2007): *Using graph concepts to understand the organization of complex systems*. IJBC 17(7), pp. 2201–2214.

[4] M. Ederer, T. Sauter, E. Bullinger, E.D. Gilles & F. Allgwer (2003): *An approach for dividing models of biological reaction networks into functional units*. Simulation 79(12), pp. 703 – 716.

[5] L. Hartwell, J. Hopfield, S. Leibler & A. Murray (1999): *From molecular to modular cell biology*. Nature 402, pp. C47 – C52.

[6] D. Irons & N. Monk (2007): *Identifying dynamical modules from genetic regulatory systems: applications to the segment polarity network*. BMC Bioinformatics 8(1), p. 413.

[7] A. S. Jarrah & R. Laubenbacher (2007): *Discrete Models of biochemical networks: the toric variety of nested canalyzing functions*. In: Algebraic Biology, AB 2007, Castle of Hagenberg, Austria, LNCS 4545. Springer, pp. 15–22.

[8] S. Kauffman (1993): *The Origins of Order*. Oxford University Press.

[9] B. N. Kholodenko, S. Schuster, J. M. Rohwer, M. Cascante & H. V. Westerhoff (1995): *Composite control of cell function: metabolic pathways behaving as single control units*. FEBS Letters 368(1), pp. 1 – 4.

[10] L. Mendoza (2006): *A network model for the control of the differentiation process in Th cells*. Biosystems 84(2), pp. 101 – 114.

[11] M.E.J. Newman (2006): *Modularity and community structure in networks*. Proc. Natl. Acad. Sci. 103(23), pp. 8577 – 8582.

[12] J. A. Papin, J. L. Reed & B. O. Palsson (2004): *Hierarchical thinking in network biology: the unbiased modularization of biochemical networks*. Trends in Biochemical Sciences 29(12), pp. 641 – 647.

[13] É. Remy, P. Ruet & D. Thieffry (2008): *Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework*. Adv. Appl. Math. 41(3), pp. 335–350.

[14] A. Richard (2009): *Positive circuits and maximal number of fixed points in discrete dynamical systems*. Discr. Appl. Math. 157, pp. 3281 – 3288.

[15] F. Robert (1986): *Discrete Iterations: A Metric Study*, Springer Series in Computational Mathematics 6. Springer.

[16] H. Siebert (2009). *Analysis of Discrete Bioregulatory Networks Using Symbolic Steady States*. Preprint 555, DFG Research Center MATHEON. `http://www.matheon.de/download.asp?File=5576_Siebert_1_09.pdf`.

[17] H. Siebert (2009): *Deriving behavior of boolean bioregulatory networks from subnetwork dynamics*. Math. Comp. Sci. 2(3), pp. 421–442.

[18] R. Thomas & R. d'Ari (1990): *Biological Feedback*. CRC Press.

[19] R. Thomas & M. Kaufman (2001): *Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits*. Chaos 11, pp. 180–195.

2nd Int. Workshop on Computational Models for Cell Processes
Eindhoven, the Netherlands, November 3, 2009
Electronic Proceedings in Theoretical Computer Science (EPTCS), volume 6
DOI: 10.4204/EPTCS.6