Katsuhisa Horimoto Georg Regensburger
Markus Rosenkranz Hiroshi Yoshida (Eds.)

Algebraic Biology

Third International Conference, AB 2008
Castle of Hagenberg, Austria, July 31–August 2, 2008
Proceedings

Springer

Volume Editors

Katsuhisa Horimoto
National Institute of Advanced Industrial Science
and Technology (AIST)
Computational Biology Research Center (CBRC)
Tokyo, Japan
E-mail: k.horimoto@aist.go.jp

Georg Regensburger
Johann Radon Institute for Computational
and Applied Mathematics (RICAM)
Austrian Academy of Sciences
Linz, Austria
E-mail: georg.regensburger@oeaw.ac.at

Markus Rosenkranz
Johann Radon Institute for Computational
and Applied Mathematics (RICAM)
Austrian Academy of Sciences
Linz, Austria
E-mail: markus.rosenkranz@oeaw.ac.at

Hiroshi Yoshida
Department of Mathematics
Kyushu University, Fukuoka, Japan
E-mail: phiroshi@math.kyushu-u.ac.jp

Library of Congress Control Number: 2008931512

CR Subject Classification (1998): F.3.1, F.4, D.2.4, I.1, J.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-85100-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-85100-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12443734 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 5147

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

This volume contains the proceedings of the Third International Conference on Algebraic Biology (AB 2008). Jointly organized by the National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, and the Research Institute for Symbolic Computation (RISC), Hagenberg, Austria, it was held from July 31 to August 2, 2008 in the Castle of Hagenberg.

Algebraic biology is an interdisciplinary forum for research on all aspects of applying symbolic computation in biology. The first conference on algebraic biology (AB 2005) was held November 28–30, 2005 in Tokyo, the second during July 2–4, 2007 in Hagenberg. The AB conference series is intended as a bridge between life sciences and symbolic computation: On the one hand, new insights in biology are found by powerful symbolic methods; on the other hand, biological problems suggest new algebraic structures and algorithms. While this profile has been established in the previous proceedings, the papers in the present volume demonstrate the continuous growth of algebraic biology.

We received 27 submissions from 14 countries (Australia, Austria, Canada, China, Colombia, France, Germany, Italy, Japan, Norway, Russia, Switzerland, UK, USA), and 14 papers were accepted for publication. Each submission was assigned to at least three Program Committee members, who carefully reviewed the papers, in many cases with the help of external referees. The reviews were discussed by the Program Committee for one week via the EasyChair conference management system.

Since the last conference, life sciences tutorials and symbolic computation tutorials have been organized as additional platforms for stimulating exchange between the communities. The idea of life sciences tutorials is to explain important problems of the area and survey past/current approaches by mathematical methods. Conversely, the goal of symbolic computation tutorials is to present the most important and successful methods in symbolic computation to experts in biology. This year we had three life sciences and three symbolic computation tutorials; three tutorial speakers submitted papers.

Furthermore, a new session—short communications with posters—was introduced for encouraging the presentation of interesting but "not-yet-polished" ideas, in particular unconventional proposals carrying the potential of creating new links between biology and symbolic computation. Despite the late announcement, six communications were submitted from five countries (Austria, France, Japan, Ukraine, USA), and five communications were accepted through a peer-viewing procedure by Program Committee members. The extended abstracts were distributed in a special booklet at the conference.

We are pleased to continue our collaboration with Springer, who agreed to publish the proceedings of AB 2007 and AB 2008 in the *Lecture Notes in Computer Science* series.

The AB Steering Committee and the organizers of the conference are grateful to the following sponsors: Austrian Grid, National Institute of Advanced Industrial Science and Technology, Radon Institute for Computational and Applied Mathematics (RICAM), RISC Software GmbH, Special Research Program SFB F013 of the Austrian Science Fund (FWF), and the Upper Austrian Government.

Our thanks are also due to all the members of the Program Committee, to the invited and tutorial speakers, to the external reviewers, and to all those who contributed to a successful and enjoyable conference.

August 2008

Bruno Buchberger
Katsuhisa Horimoto
Reinhard Laubenbacher
Bud Mishra
Georg Regensburger
Markus Rosenkranz
Hiroshi Yoshida

Conference Organization

Conference Chairs

Bruno Buchberger	Johannes Kepler University of Linz, Austria
Katsuhisa Horimoto	National Institute of Advanced Industrial Science and Technology, Japan
Reinhard Laubenbacher	Virginia Bioinformatics Institute, USA
Bud Mishra	New York University, USA

Program Chairs

Katsuhisa Horimoto	National Institute of Advanced Industrial Science and Technology, Japan
Georg Regensburger	Johann Radon Institute for Computational and Applied Mathematics, Austria
Markus Rosenkranz	Johann Radon Institute for Computational and Applied Mathematics, Austria
Hiroshi Yoshida	Kyushu University, Japan

Program Committee

Sachiyo Aburatani	National Institute of Advanced Industrial Science and Technology, Japan
Tatsuya Akutsu	Kyoto University, Japan
Hirokazu Anai	Fujitsu Laboratories Ltd., Japan
Niko Beerenwinkel	ETH Zurich, Switzerland
Armin Biere	Johannes Kepler University of Linz, Austria
Bruno Buchberger	Johannes Kepler University of Linz, Austria
Luca Cardelli	Microsoft Research, Cambridge, UK
Gautam Dasgupta	Columbia University, USA
François Fages	INRIA Rocquencourt, France
Hoon Hong	North Carolina State University, USA
Katsuhisa Horimoto	National Institute of Advanced Industrial Science and Technology, Japan
Abdul Jarrah	Virginia Bioinformatics Institute, USA
Erich Kaltofen	North Carolina State University, USA
Veikko Keränen	Rovaniemi University of Applied Sciences, Finland
Hans A. Kestler	University of Ulm, Germany
Reinhard Laubenbacher	Virginia Bioinformatics Institute, USA
Pierre Lescanne	Ecole Normale Supérieure of Lyon, France

James F. Lynch	Clarkson University, USA
Manfred Minimair	Seton Hall University, USA
Bud Mishra	New York University, USA
Eugenio Omodeo	University of Trieste, Italy
Georg Regensburger	Johann Radon Institute for Computational and Applied Mathematics, Austria
Markus Rosenkranz	Johann Radon Institute for Computational and Applied Mathematics, Austria
Stanly Steinberg	University of New Mexico, USA
Seth Sullivant	Harvard University, USA
Carolyn L. Talcott	SRI International, USA
Francis Thackeray	Transvaal Museum, Northern Flagship Institution, South Africa
Ashish Tiwari	SRI International, USA
Hiroyuki Toh	Kyushu University, Japan
Dongming Wang	Beihang University, China and UPMC-CNRS, France
Bridget S. Wilson	University of New Mexico, USA
Limsoon Wong	National University of Singapore
Kazuhiro Yokoyama	Rikkyo University, Japan
Hiroshi Yoshida	Kyushu University, Japan
Ruriko Yoshida	University of Kentucky, USA

Invited Speakers

Kiyoshi Asai	National Institute of Advanced Industrial Science and Technology, Japan
Charles Cantor	Sequenom, Inc., USA

Tutorial Speakers

Tatsuya Akutsu	Kyoto University, Japan
Armin Biere	Johannes Kepler University of Linz, Austria
François Boulier	University Lille I, France
Ken Fukuda	National Institute of Advanced Industrial Science and Technology, Japan
Tohru Natsume	Biomedicinal Information Research Center, Japan
Ashish Tiwari	SRI International, USA

Local Organization

Betina Curtis	Johannes Kepler University of Linz, Austria
Georg Regensburger	Johann Radon Institute for Computational and Applied Mathematics, Austria
Markus Rosenkranz	Johann Radon Institute for Computational and Applied Mathematics, Austria

External Reviewers

Grégory Batt
Luca Bortolussi
Christopher Brown
Franck Delaplace
Francesco Fabris
Cédric Lhoussaine
Henning Mortveit
Masahiko Nakatsui
Wei Niu
Masayuki Noro
Andrea Sgarro
Yasuhiro Suzuki
Alan Veliz-Cuba
Andreas Weber
Osvaldo Zagordi
Jun Zhang

Sponsors

Austrian Grid
National Institute of Advanced Industrial Science and Technology (AIST)
Johann Radon Institute for Computational and Applied Mathematics (RICAM)
RISC Software GmbH
Special Research Program SFB F013 of the Austrian Science Fund (FWF)
Upper Austrian Government

Table of Contents

Algorithms for Inference, Analysis and Control of Boolean Networks

Tatsuya Akutsu, Morihiro Hayashida, and Takeyuki Tamura

Bioinformatics Center, Institute for Chemical Research, Kyoto University,
Gokasho, Uji, Kyoto 611-0011, Japan
{takutsu,morihiro,tamura}@kuicr.kyoto-u.ac.jp

Abstract. Boolean networks (BNs) are known as a mathematical model of genetic networks. In this paper, we overview algorithmic aspects of inference, analysis and control of BNs while focusing on the authors' works. For inference of BN, we review results on the sample complexity required to uniquely identify a BN. For analysis of BN, we review efficient algorithms for identifying singleton attractors. For control of BN, we review NP-hardness results and dynamic programming algorithms for general and special cases.

1 Introduction

Mathematical analysis of biological networks is an important topic in bioinformatics, systems biology, and algebraic biology. For that purpose, various kinds of mathematical models have been proposed. Among them, the *Boolean network* (BN, in short) model has received much attention [16] as a model of genetic networks. BN is a very simple model: each node (e.g., gene) takes either 0 (inactive) or 1 (active) and the states of nodes change synchronously according to regulation rules given as Boolean functions. Though various aspects of BNs have been studied, this paper focuses on the following three problems.

Inference of BN: Given a part of the state transition table (which corresponds to time series data of gene expression), infer a BN that is consistent with given data.

Identification of Attractors: Given a BN, identify all attractors where attractors correspond to steady-states.

Control of BN: Given a BN with control nodes, its initial and target states, find a sequence of 0-1 vectors for control nodes which leads BN from the initial state to the target state.

These problems are considered to be fundamental and are interesting from an algorithmic viewpoint. The purpose of this review paper is not to give a comprehensive survey, but to explain the key ideas and proofs in algorithmic and mathematical results mainly obtained by the authors.

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 1–15, 2008.

2 Boolean Network

In this section, we briefly review BN [16].

A BN is represented by a set of *nodes* and a set of regulation rules for nodes, where each node corresponds to a gene if BN is regarded as a model of a genetic network. Each node takes either 0 or 1 at each discrete time t, where 1 (resp. 0) means that the corresponding gene is expressed (resp. not expressed) at time t. A regulation rule for each node is given in the form of a Boolean function and the states of nodes change synchronously. An example is given in Fig. 1. In this example, the state of node v_1 at time $t+1$ is determined by the logical AND of the states of nodes v_2 and v_3 at time t. The states of node v_2 and v_3 at time $t+1$ are determined by the state of node v_1 and the logical NOT of the state of node v_2 at time t, respectively. We use $x \wedge y$, $x \vee y$, $x \oplus y$, $\overline{x}$ to denote logical AND of x and y, logical OR of x and y, exclusive OR of x and y, and logical NOT of x, respectively. Dynamics of a BN is well-described by a state transition table and a state transition diagram shown in Fig. 1. For example, the third row of the table means that if the state of BN is $[0,1,0]$ at time t then the state will be $[0,0,0]$ at time $t+1$, and the arc from 111 to 110 in the diagram means that if the state of BN is $[1,1,1]$ at time t the state will be $[1,1,0]$ at time $t+1$.

Now we will give a formal definition of BN. A *Boolean network* $G(V,F)$ consists of a set $V = \{v_1, \ldots, v_n\}$ of nodes and a list $F = (f_1, \ldots, f_n)$ of *Boolean functions*, where a Boolean function $f_i(v_{i_1}, \ldots, v_{i_k})$ with inputs from specified nodes $v_{i_1}, \ldots, v_{i_k}$ is assigned to each node v_i. We use $IN(v_i)$ to denote the set of input nodes $v_{i_1}, \ldots, v_{i_k}$ to v_i. Each node takes either 0 or 1 at each discrete

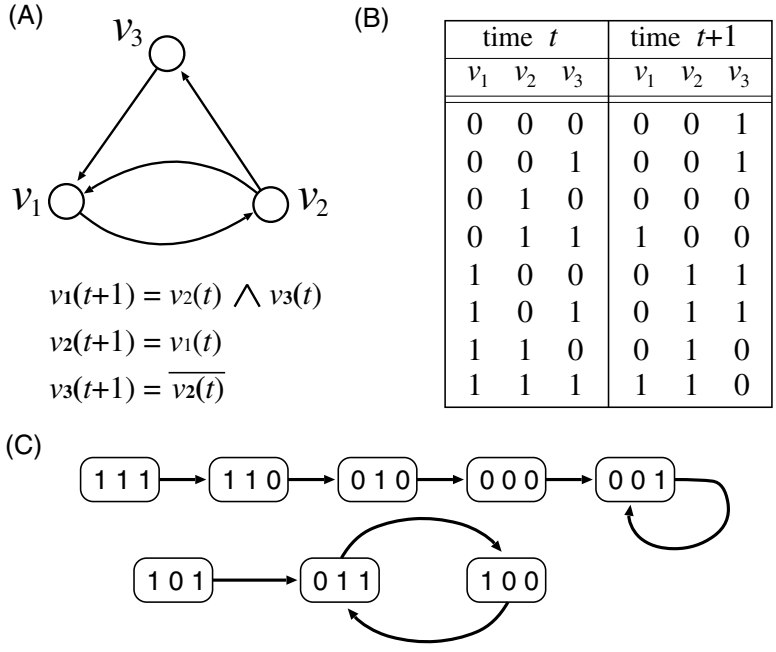


Fig. 1. Example of a Boolean network. Dynamics of BN (A) is well-described by a state transition table (B) and by a state transition diagram (C).

time t, and the state of node v_i at time t is denoted by $v_i(t)$. Then, the state of node v_i at time $t+1$ is determined by

$$v_i(t+1) = f_i(v_{i_1}(t), \ldots, v_{i_{k_i}}(t)).$$

Here we let $\mathbf{v}(t) = [v_1(t), \ldots, v_n(t)]$, which is called a *Gene Activity Profile* (GAP) at time t. We also write $v_i(t+1) = f_i(\mathbf{v}(t))$ to denote the regulation rule for v_i and $\mathbf{v}(t+1) = \mathbf{f}(\mathbf{v}(t))$ to denote the regulation rule for the whole BN. We define the set of edges E by $E = \{(v_{i_j}, v_i) | v_{i_j} \in IN(v_i)\}$. Then, $G(V, E)$ is a directed graph representing the network topology of a BN. It is worthy to mention that an edge from v_{i_j} to v_i means that v_{i_j} directly affects expression of v_i. The number of input nodes to v_i is called the *indegree* of v_i. We use K to denote the *maximum indegree* of a BN, which plays an important role in both inference and analysis of BNs.

Though BNs are deterministic, many real biological systems contain stochastic subsystems. Thus, several probabilistic extensions of BN have been proposed, which include Noisy Boolean Networks [3] and Probabilistic Boolean Networks (PBNs) [28]. Since this paper focuses on BNs, readers interested in these models are referred to [3,8,25,26,28].

3 Inference of Boolean Networks

Due to the development of DNA microarray technology, it has been made possible to observe time series data Vof expression of several thousands of genes, and thus extensive studies have been done for inferring genetic networks using these time series data. In order to infer genetic networks, mathematical models of genetic networks are usually required. In 1998, Liang et al. developed the REVEAL algorithm for inference of BNs from gene expression data [20]. Independently, Akutsu et al. studied in 1998 algorithmic strategies for identification of Boolean-like networks using gene disruption and gene overexpression [4]. By combining these two, Akutsu et al. derived a fundamental result on the number of samples (gene expression profiles) that are required to uniquely identify a BN [1]. In this section, we review this result of the sample complexity along with some algorithmic issues.

3.1 Problem Definition and Simple Inference Algorithm

Let (I^j, O^j) $(j = 1, \ldots, m)$ be a pair of expression profiles (i.e., 0-1 vectors) of $v_1, \ldots, v_n$, where I^j corresponds to a GAP at time t and O^j corresponds to a GAP at time $t+1$. I^j_i (resp. O^j_i) denotes the expression (0 or 1) of gene v_i in I^j (resp. O^j). Each pair (I^j, O^j) is called a *sample*. We say that a node v_i in a BN $G(V, F)$ is *consistent* with a sample (I^j, O^j) if $O^j_i = f_i(I^j_{i_1}, \ldots, I^j_{i_k})$ holds. We say that $G(V, F)$ is *consistent* with (I^j, O^j) if all nodes are consistent with (I^j, O^j). For a set of samples $EX = \{(I^1, O^1), (I^2, O^2), \ldots, (I^m, O^m)\}$, we say that $G(V, F)$ (resp. node v_i) is *consistent* with EX if $G(V, F)$ (resp. node v_i) is consistent with all (I^j, O^j) for $1 \leq j \leq m$. Then, the inference problem and the identification problem are defined as follows [1].

Samples

	$v_1(t)$	$v_2(t)$	$v_3(t)$	$v_1(t+1)$	$v_2(t+1)$	$v_3(t+1)$	
I^1	1	0	0	0	0	1	O^1
I^2	0	1	0	0	1	1	O^2
I^3	0	1	1	1	0	0	O^3

BN_1

$v_1(t+1) = v_3(t)$

$v_2(t+1) = v_2(t) \wedge \overline{v_3(t)}$

$v_3(t+1) = \overline{v_3(t)}$

BN_2

$v_1(t+1) = v_3(t)$

$v_2(t+1) = v_2(t) \oplus v_3(t)$

$v_3(t+1) = v_1(t) \vee \overline{v_3(t)}$

Fig. 2. Inference of BN. In this example, BNs consistent with given samples are not determined uniquely because both BN_1 and BN_2 are consistent with samples.

Definition 1 [Inference of BN]
Instance: *The number of nodes n and a set of samples $EX = \{(I^j, O^j) \mid j = 1, \ldots, m\}$,*
Problem: *Decide whether or not there exists a BN of n nodes consistent with EX and output one if it exists.*

Definition 2 [Identification of BN]
Instance: *The number of nodes n and a set of samples $EX = \{(I^j, O^j) \mid j = 1, \ldots, m\}$,*
Problem: *Decide whether or not there exists a unique BN of n nodes consistent with EX and output it if it exists.*

It is to be noted that both problems are very similar: the difference lies only in a point that the uniqueness of the solution should be verified in the identification problem.

3.2 Upper and Lower Bounds on Sample Complexity

To study the sample complexity, we consider the following quite simple algorithm for inference of BNs: for each node v_i, we generate all possible Boolean functions f_i and output a function that satisfies $O_i^j = f_i(I_{i_1}^j, \ldots, I_{i_k}^j)$ for all $j = 1, \ldots, m$. If there is no restriction, it is well known that the number of possible Boolean functions for each v_i is 2^{2^n}. If the maximum indegree is bounded by K, the number of possible Boolean functions (along with possible sets of input nodes) is at most $\binom{n}{K} 2^{2^K}$, which is a polynomial of n if K can be regarded as a constant.

Now we analyze the sample complexity: the number of samples that are required to uniquely identify a BN. It is known that BNs correspond to state-transition tables in the one-to-one manner. This means that all rows of a state-transition table are required in order to uniquely specify a BN.

Proposition 1. [1] *2^n samples are required to uniquely identify a BN if there is no restriction on a BN.*

This proposition suggests that an exponential number of gene expression profiles are required, which is non-realistic. However, if the maximum indegree is

Table 1. Explanation of Proposition 3. For $K = 1$, EX_1 satisfies the condition of the proposition, whereas EX_2 does not satisfy since $[v_2 = 0, v_3 = 0]$ does not appear.

	EX_1				EX_2			
	v_1	v_2	v_3	v_4	v_1	v_2	v_3	v_4
I_1	0	1	1	1	1	1	0	0
I_2	1	0	1	1	0	0	1	1
I_3	1	1	0	1	1	0	1	0
I_4	1	1	1	0	0	1	0	1
I_5	0	0	0	0	1	1	1	1

bounded, the situation drastically changes. First, we show a lower bound on the sample complexity.

Proposition 2. [1] *$\Omega(2^K + K \log n)$ samples are necessary in the worst case to identify a BN of maximum indegree K.*

Proof. We consider the number of mutually distinct Boolean networks. Since there are $\Omega(n^K)$ possible combinations of input nodes and 2^{2^K} possible Boolean functions per node, there are $\Omega((2^{2^K} \cdot n^K)^n)$ BNs whose maximum indegree is K.[1] Therefore, $\Omega(2^K n + nK \log n)$ bits are required to represent a BN. On the other hand, each sample gives information quantity of n bits. Therefore, $\Omega(2^K + K \log n)$ samples are required in the worst case. □

Next, we show an upper bound of the sample complexity. For that purpose, we need the following proposition (see also Table 1).

Proposition 3. [4] *If all assignments (i.e., 2^{2K} assignments) of Boolean values to all subsets of V with $2K$ nodes (i.e., $\binom{n}{2K}$ subsets) appear in I^js, the Boolean function together with input nodes for each node is determined uniquely, if it exists.*

Theorem 1. [1] *If $O(2^{2K} \cdot (2K+\alpha) \cdot \log n)$ samples (i.e., I^js) are given uniformly randomly, the following holds with probability at least $1 - \frac{1}{n^\alpha}$: there exists at most one BN of n nodes with maximum indegree $\leq K$ which is consistent with given samples.*

Proof. We derive the number of I^js satisfying the condition of Proposition 3. For that purpose, we consider the probability that the condition is not satisfied when m random I^js are given.

For any fixed set of nodes $\{v_{i_1}, \ldots, v_{i_{2K}}\}$, the probability that a sub-assignment $v_{i_1} = v_{i_2} = \cdots = v_{i_{2K}} = 1$ does not appear in one random I^j is $1 - \dfrac{1}{2^{2K}}$. Thus, the probability that $v_{i_1} = \cdots = v_{i_{2K}} = 1$ does not appear in any of

[1] The same Boolean functions may be counted multiple times. But, it does not cause a problem since we use Ω notation.

m random I^js is $(1 - \frac{1}{2^{2K}})^m$. Since the number of combinations of $2K$ nodes is less than n^{2K}, the probability that there exists a combination of $2K$ nodes for which an assignment $v_{i_1} = \cdots = v_{i_{2K}} = 1$ does not appear in any of m random I^js is at most $n^{2K} \cdot (1 - \frac{1}{2^{2K}})^m$. Since there are 2^{2K} possible assignments to $2K$ variables, the probability that the condition of Proposition 3 is not satisfied is at most $2^{2K} \cdot n^{2K} \cdot (1 - \frac{1}{2^{2K}})^m$. It is not difficult to see that $2^{2K} \cdot n^{2K} \cdot (1 - \frac{1}{2^{2K}})^m < p$ holds for $m > \ln 2 \cdot 2^{2K} \cdot (2K + 2K \log n + \log \frac{1}{p})$. Letting $p = \frac{1}{n^\alpha}$, we obtain the theorem. □

3.3 Computational Complexity Issue

The simple algorithm shown in Section 3.2 works in $O(mn^{K+1})$ time for constant K. Though it is polynomial, the degree of the polynomial becomes very high as K increases. Several efforts have been done to reduce the worst case time complexity and the practical computation time. However, it still takes long time for large K. Indeed, both the inference and identification problems are shown to be NP-hard if there is no restriction on K [2].

Though it is quite difficult to reduce the worst case time complexity, some greedy type approximation algorithms have been proposed. It is proven under the uniform distribution of samples that greedy type algorithms can identify BNs with high probability for wide-class of Boolean functions [6,12]. Furthermore, some sophisticated algorithms are proposed which work for all types of Boolean functions under the uniform distribution [23].

4 Identification of Attractors

One of extensively studied topics for BNs is analysis of the number and length of attractors in randomly generated BNs with average indegree K, where attractors correspond to steady-states. Starting from [16], a fast increase of number of attractors has been seen [7,10,27]. Although there is no conclusive result on the mean length of attractors, many researches have also been done [10,16]. Recently, several methods have been developed for efficient identification of attractors [9,13,15,30], whereas it is known that finding a singleton attractor (i.e., a fixed point) is NP-hard [21,30]. Devloo et al. developed a method using transformation to a constraint satisfaction problem [9]. Garg et al. developed a method based on Binary Decision Diagrams (BDDs) [13]. Irons developed a method that makes use of small subnetworks [15]. However, theoretical analysis of the average case time complexity was not performed in these works. We recently developed algorithms for identifying singleton attractors and small attractors and analyzed the average case time complexities of these algorithms [30]. In this section, we overview our algorithms and their analyses.

4.1 Attractors in Boolean Networks

As mentioned in Section 2, $\mathbf{v}(t+1)$ is determined from $\mathbf{v}(t)$ in a BN. Starting from an initial GAP $\mathbf{v}(0)$, a BN will eventually reach a set of global states, called an *attractor* (a directed cycle in the state transition diagram). An attractor consisting of only one global state (i.e., $\mathbf{v} = \mathbf{f}(\mathbf{v})$) is called a *singleton attractor*, which corresponds to a fixed point. Otherwise, it is called a *cyclic attractor* with period p if it consists of p global states (i.e., $\mathbf{v}^1 = \mathbf{f}(\mathbf{v}^p) = \mathbf{f}(\mathbf{f}(\mathbf{v}^{p-1})) = \cdots = \mathbf{f}(\mathbf{f}(\cdots\mathbf{f}(\mathbf{v}^1)\cdots)))$. The set of all GAPs that eventually evolve into the same attractor is called the *basin of attraction*. Different basins of attraction correspond to different connected components in the state transition diagram, and each connected component contains exactly one directed cycle. For example, in Fig. 1, 001 is a singleton attractor, $\{011, 100\}$ is a cyclic attractor with period 2, and $\{111, 110, 010, 000, 001\}$ are the basin of the singleton attractor 001.

In this paper, the attractor identification problem is defined as a problem of enumerating all attractors for a given BN. However, it is very difficult to find attractors with long periods. Thus, we focus on identification of singleton attractors and identification of attractors with period at most some given threshold p_{max}. These problems are defined as below, where the *singleton attractor identification problem* corresponds to the case of $p_{max} = 1$.

Definition 3 [Identification of Attractors in BN]
Instance: *A BN and the maximum length of period* p_{max}*,*
Problem: *Enumerate all attractors with period at most* p_{max}*.*

4.2 Simple Recursive Algorithm and Its Average Case Analysis

We developed several algorithms for identifying singleton attractors and cyclic attractors with short periods [30]. For that purpose, we proposed a very simple algorithm, which is referred to as the *basic recursive algorithm* in this paper.

The number of singleton attractors in a BN depends on the regulatory rules of the network. If the rules are given as $v_i(t+1) = v_i(t)$ for all i, the number of singleton attractors is 2^n. Thus, it would take $O(2^n)$ time in the worst case if all the singleton attractors are to be identified. On the other hand, it is known that the average number of singleton attractors is 1 regardless of the number of genes n and the maximum indegree K [22]. The basic recursive algorithm was designed based on these facts. It examines much smaller number of states than 2^n in the average case.

In the algorithm, a partial GAP (i.e., profile with m ($< n$) genes) is extended one by one towards a complete GAP (i.e., singleton attractor), according to a given gene ordering (i.e., a random gene ordering). If it is found that a partial GAP cannot be extended to a singleton attractor, the next partial GAP is examined. The pseudocode of this algorithm is given below, where this procedure is invoked with $m = 1$.

Procedure *IdentSingletonAttractor*(v, m)
  if $m = n + 1$
  then Output $v_1(t), v_2(t), \cdots, v_n(t)$, **return**;
  for $b = 0$ **to** 1 **do**
    $v_m(t) := b$;
    if it is found that $f_j(\mathbf{v}(t)) \neq v_j(t)$ for some $j \leq m$
    then continue
    else *IdentSingletonAttractor*$(v, m + 1)$;
  return;

This algorithm extends a partial GAP by one gene at a time in a recursive manner. At the m-th recursive step, the states of the first $m - 1$ genes (i.e., a partial GAP) are already determined. Then, the algorithm extends the partial GAP by letting $v_m(t) = 0$. If $v_j(t + 1) = v_j(t)$ holds or the value of $v_j(t + 1)$ is not determined for each $j = 1, \ldots, m$, the algorithm proceeds to the next recursive step. That is, if there is a possibility that the current partial GAP can be extended to a singleton attractor, it goes to the next recursive step. Otherwise, it extends the partial GAP by letting $v_m(t) = 1$ and executes a similar procedure. After examining $v_m(t) = 0$ and $v_m(t) = 1$, the algorithm returns to the previous recursive step. Since the number of singleton attractors is small in most cases, it is expected that the algorithm does not examine many partial GAPs with large m. The average case time complexity is estimated as follows [30].

Assume that we have tested the first m out of n genes, where $m \geq K$. For all $i \leq m$, $v_i(t) \neq v_i(t + 1)$ holds with probability

$$P(v_i(t) \neq v_i(t+1)) \;=\; 0.5 \cdot \frac{\binom{m}{k_i}}{\binom{n}{k_i}} \;\approx\; 0.5 \cdot (\frac{m}{n})^{k_i} \;\geq\; 0.5 \cdot (\frac{m}{n})^{K},$$

where we assume that Boolean functions of k_i inputs are selected at uniformly random. If $v_i(t) \neq v_i(t + 1)$ holds for some $i \leq m$, the algorithm cannot go to the next recursive level. Therefore, the probability that the algorithm examines the $(m + 1)$-th gene is no more than

$$[1 - P(v_i(t) \neq v_i(t+1))]^m \;=\; [1 - 0.5 \cdot (\frac{m}{n})^K]^m.$$

Thus, the number of recursive calls executed for the first m genes is at most

$$f(m) \;=\; 2^m \cdot [1 - 0.5 \cdot (\frac{m}{n})^K]^m.$$

Let $s = \frac{m}{n}$, and $F(s) = [2^s \cdot (1 - 0.5 \cdot s^K)^s]^n = [(2 - s^K)^s]^n$. The average case time complexity is estimated by computing the maximum value of $F(s)$. Though an additional $O(nm)$ factor is required, it can be ignored since $O(n^2 a^n) \ll O((a + \epsilon)^n)$ holds for any $a > 1$ and $\epsilon > 0$.

Since we want to analyze the time complexity as a function of n, we only need to compute the maximum value of the function $g(s) = (2 - s^K)^s$, which can be

Table 2. Theoretically estimated average case time complexities of basic, outdegree-based, and BFS-based algorithms for the singleton attractor detection problem [30]

K	2	3	4	5
basic	1.35^n	1.43^n	1.49^n	1.53^n
outdegree-based	1.19^n	1.27^n	1.34^n	1.41^n
BFS-based	1.16^n	1.27^n	1.35^n	1.41^n

obtained by a simple numerical calculations for fixed K. Then, the average case time complexity of the algorithm can be estimated as $O((\max(g))^n)$. The average case time complexities for $K = 2, \ldots, 5$ are listed in the first row of Table 2. It should be noted that the naive exhaustive search-based algorithm takes at least $O(2^n)$ time. Thus, the basic recursive algorithm is much faster than the naive algorithm for small K.

We obtained variants of this basic recursive algorithm by sorting nodes before invoking the recursive procedure [30]. In particular, we used the orderings of nodes according to the outdegree and BFS (breadth-first search). For these algorithms, we obtained theoretical estimates of the average case time complexity (see Table 2). We also performed computational experiments to confirm these theoretical results (it is to be noted that some approximations were included in theoretical analyses). As a result, good agreement was observed. We also extended the basic recursive algorithm for identifying cyclic attractors with short period [30]. Though the extended algorithm is not efficient compared with those in Table 2, it still works in $o(2^n)$ time in the average case.

4.3 Issues on the Worst Case Time Complexity

We have considered the average case time complexity in the above. However, it is also very important to consider the worst case time complexity. We have shown that the singleton attractor detection problem (i.e., decide whether or not there exists a singleton attractor) can be solved in $o(2^n)$ time for constant K by a reduction to the satisfiability problem for CNF (conjunctive normal form). We have also shown that the singleton attractor detection problem can be solved in $o(2^n)$ time for general K if Boolean functions are restricted to AND/OR of literals [29]. However, no $o(2^n)$ time algorithm is known for more general cases of the singleton attractor detection problem and thus development of such an algorithm is left as an open problem.

5 Control of Boolean Networks

One of the major goals of systems biology is to develop a control theory for biological systems [17,18]. Development of such a control theory is important both from a theoretical viewpoint and from a practical viewpoint. From a theoretical viewpoint, biological systems are complex and contain highly non-linear subsystems and thus existing methods in control theory cannot be directly applied to

control of biological systems. Therefore, it is quite interesting to develop theory and methods for control of biological systems. From a practical viewpoint, control of cells may be useful for systems-based drug discovery and cancer treatment [17,18]. Since BNs are highly non-linear systems, it is reasonable to try to develop methods for control of BNs.

Datta et al. proposed a method for finding a control strategy for PBN [8], from which many extensions followed [11,25,26]. In their approach, it is assumed that states of some nodes can be externally controlled and the objective is to find a sequence of control actions with the minimum cost that leads to a desirable state of a network. Their approach is based on the theory of Markov chains and makes use of the classical technique of dynamic programming. Since BNs are special cases of PBNs, their methods can also be applied to control of BNs. However, it is required in their methods to handle exponential size matrices and thus their methods can only be applied to small biological systems. Therefore, it is reasonable to ask how difficult it is to find control strategies for BNs. We showed that finding control strategies for BNs is NP-hard [5], which means that there is no polynomial time algorithm unless P=NP [14]. On the other hand, we showed that this problem can be solved in polynomial time if BN has a tree structure. In this section, we review these results along with the essential idea of [8].

5.1 Definition of the Control Problem

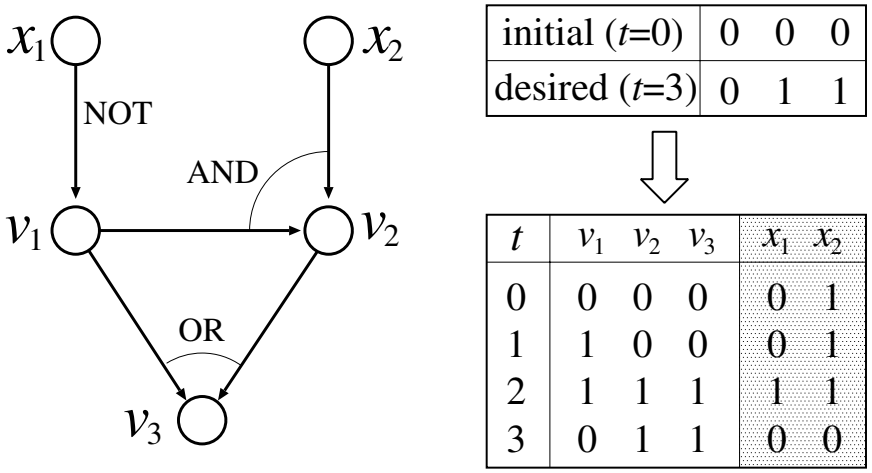
Here we give a formal definition of the problem of finding control strategies for BNs (Control of BN) [5].

In Control of BN, it is assumed that there exist two types of nodes: *internal nodes* and *external nodes*, where internal nodes correspond to usual nodes (i.e., genes) in BN and external nodes correspond to control nodes. Let a set V of $n+m$ nodes be $V = \{v_1, \ldots, v_n, v_{n+1}, \ldots, v_{n+m}\}$, where $v_1, \ldots, v_n$ are internal nodes and $v_{n+1}, \ldots, v_{n+m}$ are external nodes. For convenience, we use x_i to denote an external node v_{n+i}. Then, states of internal nodes ($v_i(t+1)$ for $i = 1, \ldots, n$) are determined by

$$v_i(t+1) = f_i(v_{i_1}(t), \ldots, v_{i_{k_i}}(t)),$$

where each v_{i_k} is either an internal node or an external node. Here, we let $\mathbf{v}(t) = [v_1(t), \ldots, v_n(t)]$ and $\mathbf{x}(t) = [x_1(t), \ldots, x_m(t)]$. We can describe the dynamics of a BN by $\mathbf{v}(t+1) = \mathbf{f}(\mathbf{v}(t), \mathbf{x}(t))$, where $\mathbf{x}(t)$s are determined externally. Then, Control of BN is defined as follows (see also Fig. 3) [5].

Definition 4 (Control of BN)
Instance: *A BN, an initial state of the network for internal nodes* $\mathbf{v}^0$*, and the desired state of the network* $\mathbf{v}^M$ *at the* M*-th time step,*
Problem: *Find a sequence of 0-1 vectors* $\langle \mathbf{x}(0), \ldots, \mathbf{x}(M) \rangle$ *such that* $\mathbf{v}(0) = \mathbf{v}^0$ *and* $\mathbf{v}(M) = \mathbf{v}^M$*. If there does not exist such a sequence, "None" should be the output.*



initial (t=0)	0	0	0
desired (t=3)	0	1	1

t	v_1	v_2	v_3	x_1	x_2
0	0	0	0	0	1
1	1	0	0	0	1
2	1	1	1	1	1
3	0	1	1	0	0

Fig. 3. Example of Control of BN. In this problem, given initial and desired states of internal nodes (v_1, v_2, v_3), it is required to compute a sequence of states of external nodes (x_1, x_2) leading to the desired state.

5.2 Dynamic Programming Algorithms for Control of BNs

As mentioned before, Datta et al. proposed a dynamic programming based method for finding a control strategy for PBN [8], which can also be applied to BN. Here, we briefly review their method in the context of BN.

We use a table $D[b_1, \ldots, b_n, t]$, where each entry takes either 0 or 1. $D[b_1, \ldots, b_n, t]$ takes 1 if there exists a control sequence $\langle \mathbf{x}(t), \ldots, \mathbf{x}(M) \rangle$ which leads to the target state $\mathbf{v}^m$ beginning from the state $[b_1, \ldots, b_n]$ at time t. This table is computed from $t = M$ to $t = 0$ by using the following procedure:

$$D[b_1, \ldots, b_n, M] = \begin{cases} 1, \text{ if } [b_1, \ldots, b_n] = \mathbf{v}^M, \\ 0, \text{ otherwise,} \end{cases}$$

$$D[b_1, \ldots, b_n, t-1] = \begin{cases} 1, \text{ if there exists } (\mathbf{c}, \mathbf{x}) \text{ such that } D[c_1, \ldots, c_n, t] = 1 \\ \quad \text{and } \mathbf{c} = \mathbf{f}(\mathbf{b}, \mathbf{x}), \\ 0, \text{ otherwise,} \end{cases}$$

where $\mathbf{b} = [b_1, \ldots, b_n]$ and $\mathbf{c} = [c_1, \ldots, c_n]$. Then, there exists a desired control sequence if and only if $D[a_1, \ldots, a_n, 0] = 1$ holds for $\mathbf{v}^0 = [a_1, \ldots, a_n]$. Once the table is constructed, a desired control sequence can be obtained using the *traceback technique*, which is a standard technique in dynamic programming.

In this method, the size of table $D[b_1, \ldots, b_n, t]$ is clearly $O(M \cdot 2^n)$. Moreover, we should examine pairs of $O(2^n)$ internal states and $O(2^m)$ external states for each time t. Thus, it requires $O(M \cdot 2^{n+m})$ time excluding the time for calculation of Boolean functions. Therefore, the dynamic programming algorithm in [8] is an exponential time algorithm.

As shown in the next subsection, control of BN is NP-hard, which suggests that exponential time is inevitable in a general case. However, we may be able to develop polynomial time algorithms for special cases. We developed such an algorithm for the case where the network has a tree structure (i.e., the graph

is connected and there is no cycle). Since the algorithm is a bit complicated, we show here a simple algorithm for the case where the network has a rooted tree structure (i.e., all paths are directed from leaves to the root). In order to compute a control strategy, we employ dynamic programming. Though dynamic programming is also employed in the above, it is used here in a significantly different way.

In order to apply dynamic programming, we define a table $S[v_i, t, b]$ as below, where v_i is a node in a BN, t is a time step and b is a Boolean value (i.e., 0 or 1). Here $S[v_i, t, b]$ takes 1 if there exists a control sequence (up to time t) that makes $v_i(t) = b$.

$$S[v_i, t, 1] = \begin{cases} 1, \text{ if there exists } \langle \mathbf{x}(0), \ldots, \mathbf{x}(t) \rangle \text{ such that } v_i(t) = 1, \\ 0, \text{ otherwise.} \end{cases}$$

$$S[v_i, t, 0] = \begin{cases} 1, \text{ if there exists } \langle \mathbf{x}(0), \ldots, \mathbf{x}(t) \rangle \text{ such that } v_i(t) = 0, \\ 0, \text{ otherwise.} \end{cases}$$

Then, $S[v_i, t, 1]$ can be computed by the following dynamic programming procedure.

$$S[v_i, t+1, 1] = \begin{cases} 1, \text{ if there exists } [b_{i_1}, \ldots, b_{i_k}] \text{ such that } f_i(b_{i_1}, \ldots, b_{i_k}) = 1 \\ \quad \text{holds and } S[v_{i_j}, t, b_{i_j}] = 1 \text{ holds for all } j = 1, \ldots, k, \\ 0, \text{ otherwise.} \end{cases}$$

$S[v_i, t, 0]$ can be computed in a similar way. It should be noted that each leaf is either a constant node or an external node. For a constant node, either $S[v_i, t, 1] = 1$ and $S[v_i, t, 0] = 0$ hold for all t, or $S[v_i, t, 1] = 0$ and $S[v_i, t, 0] = 1$ hold for all t. For an external node, $S[v_i, t, 1] = 1$ and $S[v_i, t, 0] = 1$ hold for all t. Since the size of table $S[v_i, t, b]$ is $O((n + m)M)$, the above dynamic programming algorithm works in polynomial time where we assume that the value of each Boolean function can be computed in polynomial time. A desired control sequence can also be obtained from the table in polynomial time using the traceback technique. This algorithm was extended for BNs with general tree structures [5].

Theorem 2. [5] *Control of BN can be solved in polynomial time if BN has a tree structure.*

5.3 NP-Hardness Results on Control of BNs

As mentioned in the above, the dynamic programming algorithm for control of general BNs takes exponential time and thus is not efficient. However, the following theorem suggests that it is impossible (under the assumption of P≠NP) to develop a polynomial time algorithm for the general case.

Theorem 3. [5] *Control of BN is NP-hard.*

Proof. We present a simple polynomial time reduction from 3SAT [14]. Let $y_1, \ldots, y_N$ be Boolean variables (i.e., 0-1 variables). Let $c_1, \ldots, c_L$ be a set of

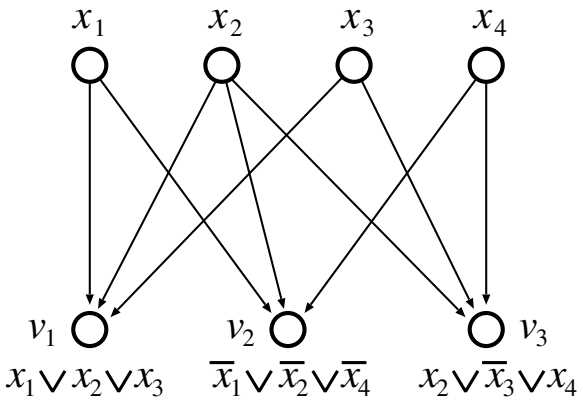


Fig. 4. Reduction from 3SAT to Control of BN. An instance of 3SAT $\{y_1 \vee y_2 \vee y_3, \overline{y_1} \vee \overline{y_2} \vee \overline{y_4}, y_2 \vee \overline{y_3} \vee y_4\}$ is transformed into this instance of Control of BN.

clauses over $y_1, \ldots, y_N$, where each clause is a logical OR of at most three literals. It should be noted that a literal is a variable or its negation (logical NOT). Then, 3SAT is a problem of asking whether or not there exists an assignment of 0-1 values to $y_1, \ldots, y_N$ which satisfies all the clauses (i.e., the values of all clauses are 1).

From an instance of 3SAT, we construct a BN as follows (see also Fig. 4). We let the set of nodes $V = \{v_1, \ldots, v_L, x_1, \ldots, x_N\}$ where each v_i corresponds to c_i and each x_j corresponds to y_j. Suppose that $f_i(y_{i_1}, \ldots, y_{i_3})$ is a Boolean function assigned to c_i in 3SAT. Then, we assign $f_i(x_{i_1}, \ldots, x_{i_3})$ to v_i in the BN. Finally, we let $M = 1$, $\mathbf{v}^0 = [0, 0, \ldots, 0]$ and $\mathbf{v}^M = [1, 1, \ldots, 1]$.

Then, it is straight-forward to see that there exists a control strategy $\langle \mathbf{x}(0), \mathbf{x}(1) \rangle$ which makes $\mathbf{v}(1) = [1, 1, \ldots, 1]$ if and only if there exists an assignment which satisfies all the clauses. Since the reduction can be done in linear time, Control of BN is NP-hard. □

It is also shown in [5] that the control problem remains NP-hard even for BNs having very restricted network structures. Especially, it is shown that it remains NP-hard if the network contains only one control node and all the nodes are OR or AND nodes (i.e., there is no negative control). However, it is unclear whether the control problem is NP-hard or can be solved in polynomial time if a BN contains a fixed number of directed cycles or loops (it is unclear even for the case of two cycles). Deciding the complexity of such a special case is left as an open problem.

Though we have shown negative results, NP-hardness does not necessarily mean that we cannot develop practical algorithms which work efficiently in the average case. For that purpose, an approximate finite-horizon optimal control has been introduced [24] and a heuristic method based on Q-learning algorithm for approximating the optimal infinite-horizon control policy has been proposed [11]. However, application of these approaches is still limited to small networks. Recently, Langmund and Jha proposed a method using techniques from the field of *model checking* [19]. They reported that the method could be applied to a BN model of embryogenesis in *D. melanogaster* with 15,360 Boolean variables.

6 Concluding Remarks

We have overviewed algorithmic aspects of inference, analysis and control of BNs with focusing on the authors' works. We understand that there is a criticism that BN is too simple as a model of genetic networks and/or other biological networks. However, studies on BNs may provide some insights into other models. At least, negative results should hold for more general models. Some ideas in positive results may also be useful for theoretical analyses and design of algorithms for more general models. For instance, Mochizuki performed theoretical analysis on the number of steady states in some continuous biological networks that are based on nonlinear differential equations [22]. The core part of the analysis is done in a combinatorial manner and is very close to that for BNs. Therefore, it is worthy to study extensions of the methods and results on BNs for more general models.

References

1. Akutsu, T., Miyano, S., Kuhara, S.: Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In: Proc. Pacific Symposium on Biocomputing 1999, pp. 17–28 (1999)
2. Akutsu, T., Miyano, S., Kuhara, S.: Algorithms for identifying Boolean networks and related biological networks based on matrix multiplication and fingerprint function. Journal of Computational Biology 7, 331–343 (2000)
3. Akutsu, T., Miyano, S., Kuhara, S.: Inferring qualitative relations in genetic networks and metabolic pathways. Bioinformatics 16, 727–734 (2000)
4. Akutsu, T., Kuhara, S., Maruyama, O., Miyano, S.: Identification of genetic networks by strategic gene disruptions and gene overexpressions under a boolean model. Theoretical Computer Science 298, 235–251 (2003)
5. Akutsu, T., Hayashida, M., Ching, W.-K., Ng, M.K.: Control of Boolean networks: Hardness results and algorithms for tree-structured networks. Journal of Theoretical Biology 244, 670–679 (2007)
6. Arpe, J., Reischuk, R.: When does greedy learning of relevant attributes succeed? In: Lin, G. (ed.) COCOON. LNCS, vol. 4598, pp. 296–306. Springer, Heidelberg (2007)
7. Bilke, S., Sjunnesson, F.: Number of attractors in random Boolean networks. Physical Review E 72, 016110 (2005)
8. Datta, A., Choudhary, A., Bittner, M.L., Dougherty, E.R.: External control in Markovian genetic regulatory networks. Machine Learning 52, 169–191 (2003)
9. Devloo, V., Hansen, P., Labbé, M.: Identification of all steady states in large networks by logical analysis. Bulletin of Mathematical Biology 65, 1025–1051 (2003)
10. Drossel, B., Mihaljev, T., Greil, F.: Number and length of attractors in a critical Kauffman model with connectivity one. Physical Review Letters 94, 088701 (2005)
11. Faryabi, B., Datta, A., Dougherty, E.R.: On approximate stochastic control in genetic regulatory networks. IET Systems Biology 1, 361–368 (2007)
12. Fukagawa, D., Akutsu, T.: Performance analysis of a greedy algorithm for inferring Boolean functions. Information Processing Letters 93, 7–12 (2005)

13. Garg, A., Xenarios, I., Mendoza, L., DeMicheli, G.: An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 62–76. Springer, Heidelberg (2007)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., New York (1979)
15. Irons, D.J.: Improving the efficiency of attractor cycle identification in Boolean networks. Physica D 217, 7–21 (2006)
16. Kauffman, S.A.: The Origins of Order: Self-organization and Selection in Evolution. Oxford Univ. Press, New York (1993)
17. Kitano, H.: Computational systems biology. Nature 420, 206–210 (2002)
18. Kitano, H.: Cancer as a robust system: implications for anticancer therapy. Nature Reviews Cancer 4, 227–235 (2004)
19. Langmead, C.J., Jha, S.K.: Symbolic approaches for finding control strategies in Boolean networks. In: Proc. 6th Asia-Pacific Bioinformatics Conference, pp. 307–319. Imperial College Press, London (2008)
20. Liang, S., Fuhrman, S., Somogyi, R.: REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. In: Proc. Pacific Symposium on Biocomputing 1998, pp. 18–29 (1998)
21. Milano, M., Roli, A.: Solving the safistiability problem through Boolean networks. In: Lamma, E., Mello, P. (eds.) AI*IA 1999. LNCS (LNAI), vol. 1792, pp. 72–93. Springer, Heidelberg (2000)
22. Mochizuki, A.: An analytical study of the number of steady states in gene regulatory networks. Journal of Theoretical Biology 236, 291–310 (2005)
23. Mossel, E., O'Donnell, R., Servedio, R.A.: Learning functions of k relevant variables. Journal of Computer and System Sciences 69, 421–434 (2004)
24. Ng, M.K., Zhang, S.-Q., Ching, W.-K., Akutsu, T.: A control model for Markovian genetic regulatory network. Transactions on Computational Systems Biology V, 36–48 (2006)
25. Pal, R., Datta, A., Bittner, M.L., Dougherty, E.R.: Intervention in context-sensitive probabilistic Boolean networks. Bioinformatics 21, 1211–1218 (2005)
26. Pal, R., Datta, A., Bittner, M.L., Dougherty, E.R.: Optimal infinite-horizon control for probabilistic Boolean networks. IEEE Transactions on Signal Processing 54, 2375–2387 (2006)
27. Samuelsson, B., Troein, C.: Superpolynomial growth in the number of attractors in kauffman networks. Physical Review Letters 90, 098701(2003)
28. Shmulevich, I., Dougherty, E.R., Kim, S., Zhang, W.: Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics 18, 261–274 (2002)
29. Tamura, T., Akutsu, T.: An improved algorithm for detecting a singleton attractor in a Boolean network consisting of AND/OR nodes. In: Proceedings of the 3rd International Conference on Algebraic Biology (to appear)
30. Zhang, S.-Q., Hayashida, M., Akutsu, T., Ching, W.-K., Ng, M.K.: Algorithms for finding small attractors in Boolean networks. EURASIP Journal on Bioinformatics and Systems Biology 2007, 20180 (2007)

Tutorial on Model Checking: Modelling and Verification in Computer Science

Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University, Linz, Austria

Abstract. This paper serves as background material for an invited tutorial on model checking given at the Third International Conference on Algebraic Biology (AB 2008). The intended audience of the tutorial were researchers in natural science, particularly life science, but this paper may also serve as a light-weight introduction into model checking techniques in general.

1 Introduction

In that part of computer science which is concerned with constructing systems, modelling usually has a different flavor than modelling in natural science. The artifacts resp. systems engineered by computer scientists usually do have precise mathematical semantics and work according to these abstract semantics. Therefore computer science allows to use precise models, which are conservative abstractions:

> If a model of a computer science system has a certain property,
> then the real system has this property as well.

This statement is of course incorrect if the system is not interpreted on an abstract level. For instance, if a processor on which a program runs has a defect or even just the compiler that produced the machine code from the original program, then the system as a whole does not have to be correct and it may violate the desired property, even though the program, e.g. the computer science artifact, is correct.

Nevertheless, computer science is able to *prove* resp. *check* properties of real systems, because these systems, in the form of programs or circuit designs, are still abstract. Using automatic techniques for checking properties of computer science systems is the purpose of *model checking*.

Research in model checking is centered around algorithmic aspects, particularly on how to implement model checkers, or related questions, such as which specification and modelling languages can be model checked efficiently. In this paper we focus on those approaches that turn out be successful in practice.

2 Modelling

An important question is where the models come from. There are two radically different ways to obtain models. In the first scenario a system to be built is

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 16–21, 2008.

modelled using some high-level and abstract modelling language. Usually only one aspect of the final not yet existing system is modelled, such as synchronization of parallel components. Then the model can be analyzed through simulation, i.e. by testing it, or by automatically checking certain properties with the help of a model checker.

After the designer has a good understanding of all the aspects of the model, he makes sure that the model is not overly simplistic and also does not contain any fundamental flaws. Then the model is typically thrown away and the system reimplemented in detail from scratch, usually in a totally different but more concrete language. This is a proven technique in industry, particular since the idea of exploring the design space through an executable model, which can be simulated, is very useful even without using model checking techniques.

In the second scenario, model checkers are applied to concrete systems, such as hardware designs, device drivers, or in general software, described in concrete implementation respectively system description languages. The point is that the description of the system in this scenario is detailed enough, even though it is still a model of the real system, that automatic techniques, particularly compilers, can be used to generate the final product.

Originally, models were finite state. This restriction, at least in principle, allows model checkers to be fully automatic. Then model checking terminates, and it either determines that a property holds on the model or the model checker provides a witness in form of a trace that shows that the property is violated. But due to the state-explosion-problem, which says that the number of states in a model is exponential in the size of its description, it may just take too much time to explore all these states and typically also too much space. Much progress has been made over the years to ameliorate this situation and improve scalability of model checking.

Some of the research in model checking also went into the other direction, and lifted the finiteness restriction. There are various forms of infinite systems, for which theoretical results are available and practical applications exist. One direction is to allow continuous variables in the data domain, another to model continuous time. A third direction is to add probability, and a fourth to parameterize the size or the number of components. Another extension is to replace finite state automata by push down automata. All of these extension have in common that model checking only remains decidable, and thus an automatic almost push-button technology, if the class of models allow finite abstractions in some way or another.

3 History

Model checking was invented more than 25 years ago in the early 80'ties by E. Clarke and A. Emerson [5] and independently by J. Queille and J. Sifakis [14]. There was a workshop [16] affiliated to the Federated Conference on Logic in Computer Science (FLOC'06) dedicated to this anniversary. Beside the proceedings [16] of this workshop, another reference for model checking research is *the*

model checking book [7]. More recently Clarke, Emerson and Sifakis won the 2007 Turing Award for their pioneering work on model checking.

From a historical perspective it is probably important to mention that initially these ideas were not immediately embraced by the formal verification community eagerly, which at that time was still mainly focused on theorem proving techniques. The main argument was that model checking, as it was described initially would only work on tiny models and thus would not scale.

On the one hand this argument is still valid, particularly if the goal is to produce a fully verified concrete system. Without additional manual and automatic abstraction techniques, model checking alone will fail in such an endeavor due to the large number of system states, that have to be explored.

On the other hand model checking has been very successful in providing complementary techniques to simulation and testing in order to *partially* verify concrete systems. Particularly in circuit design, where testing costs and also costs for defects that slip through testing are very high, model checking is applied routinely nowadays. It was shown recently, that hybrid techniques that combine model checking with automated theorem proving, can check much larger systems than each technique alone, even in checking properties of device drivers in an industrial operating system [1], for which this hybrid technique is actually used routinely now as well.

Finally, using model checking checking for pure models, e.g. the first scenario discussed in the previous section, will always be beneficial for systems with complex interactions, such as communication protocols etc.

4 Temporal Logic

Another aspect where model checking differs from other formal approaches is the choice of the specification languages, which are used to describe properties. In essence model checking is concerned with sequential or temporal behavior of systems. This kind of properties are particularly important for reactive, distributed or parallel systems and typically are described in temporal logic. A. Pnueli [12] is considered to be the first who noticed that specifications of such concurrent systems would benefit from using temporal logic.

In its simplest form temporal logic allows to specify two kinds of temporal behavior. A *safety* property states that a certain error or catastrophic state is not reachable. A dual formulation is, that a safety property holds, if all reachable states fulfill a certain invariant, which is valid initially and remains valid no matter how the system evolves. In terms of programming languages an assertion is a typical simple safety property.

More complex temporal specifications are *liveness* properties, also often referred to as *progress* properties. They are used to specify reactiveness, progress or non-starvation etc. A typical example are request/acknowledge properties, for instance calling the elevator (request), will eventually lead to the elevator doors to open (acknowledge). Another example is that a system after powering up will eventually end up in a properly "initialized" state, no matter in what

configuration it started. Liveness usually only makes sense in combination with additional fairness assumptions, for instance, that each component is allowed to have its turn infinitely often.

There are various flavors of temporal logic, most notably computation tree logic (CTL) and linear temporal logic (LTL). Related formalisms, such as omega-regular languages and μ-calculus are also used quite frequently. More information on these formalisms can be found in [7]. There are also standardized temporal logics in industry, e.g. the property specification logic (PSL).

5 Technology

At the core of model checking are algorithms that implement state space traversal. The reachable state space is traversed to find error states that violate safety properties, or to find cyclic paths on which no progress is made as counter example for liveness properties. In most cases state space traversal can be reformulated as fix-point computation, which for certain temporal logics is the only way to describe model checking algorithms.

Initially, model checkers worked on an explicit state space representation. Each state of the system is represented in the computer explicitly and typically stored in a large hash table. The size of the hash table is closely related to the number of reachable states of the system and thus computer memory became the bottle neck.

There are various techniques to improve space consumption in explicit state model checkers. The most important one is partial-order reduction, which is particularly useful for asynchronous models such as loosely coupled software components, petri-net models etc. Also, if the model is symmetric, these symmetries can be factored out during state space traversal. Finally, if the number of states is still too large to be handled, the idea of bit-state hashing trades scalability for completeness, e.g. model checking becomes a falsification technique, similar to traditional testing, but unable to prove correctness.

The most successful explicit model checker is the SPIN model checker. Its main author, G. Holzmann, received the ACM Software System Award in 2002 for his work on SPIN. His latest book [9] on SPIN is probably the best reference to start learning more about explicit state model checking and its optimizations mentioned above. There is a yearly workshop series on SPIN as well, which has more recent results.

For infinite models there are no explicit methods. States have to be represented symbolically. But even for finite models it is possible to represent states, more precisely set of states, symbolically. The goal is to overcome the state-explosion-problem.

In the mid 80'ties Randy Bryant presented reduced ordered binary decision diagrams (BDDs) as a new data structure for symbolically representing and manipulating boolean functions efficiently [4]. This paper has turned out to be one of the most cited papers in computer science.

Also at CMU a couple of years later E. Clarke and his students picked up this idea and showed that BDDs can also be used to represent set of states with BDDs and more importantly, how state space traversal techniques based on fix-point computations can be implemented efficiently using BDD operations. This break through in the early 90'ties is documented in K. McMillan's thesis [10] which also contains a detailed description of his model checker, the symbolic model verifier SMV. One can even argue that the event of symbolic model checking started a renaissance of formal verification in general now with a focus on real applications.

The research in model checking of the 90'ties produced many refinements of symbolic model checking, both in the core algorithms for finite systems, but also in applying similar techniques to infinite systems. The literature is too large to be listed here explicitly. Please refer to the model checking book [7] and in general to the proceedings of the main conference in model checking research, the conference of computer-aided verification (CAV).

In the late 90'ties it was observed that techniques for boolean satisfiability checking, i.e. SAT solvers, could handle much larger formulas than BDDs, and again researchers at CMU came up with the idea of bounded model checking (BMC) [2], which applies SAT solvers to the model checking problem. BMC in its basic form is a falsification technique, at least in practice, i.e. it trades completeness for scalability. However, follow-up work on BMC improved this situation, particular the work on k-induction [15] and interpolation [11]. A survey on these and other related techniques based on using SAT for model checking can be found in [13].

The improvement in SAT solver technology even accelerated in the last 8 years after the introduction of BMC and made model checking much more useful in industry. SAT and its extension of satisfiability modulo theories (SMT) are *the* working horse in almost all state-of-the-art applications of formal methods in industry. Again as example consider [1], which uses SMT solvers to automatically generate abstractions [8] for actual device drivers, which are then checked by a symbolic model checker. If the abstraction is too coarse the abstraction is refined [6], again with the help of SMT solvers. For more information on SAT see the Handbook of Satisfiability [3] and recent proceedings of the yearly SAT conference.

6 Conclusion

This short note represents a personal interpretation of the first 27 years of model checking from a computer science perspective. It summarizes the historical development and gives pointers to recent important results. We hope that we contributed to spread these ideas further across the boundaries of computer science.

References

1. Ball, T., Rajamani, S.: Automatically validating temporal safety properties of interfaces. In: Dwyer, M.B. (ed.) SPIN 2001. LNCS, vol. 2057. Springer, Heidelberg (2001)

2. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Cleaveland, W.R. (ed.) ETAPS 1999 and TACAS 1999. LNCS, vol. 1579. Springer, Heidelberg (1999)
3. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. IOS Press, Amsterdam (to be published, 2008)
4. Bryant, R.: Graph Based Algorithms for Boolean Function Manipulation. IEEE Trans. on Computers C(35) (1986)
5. Clarke, E., Emerson, E.: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, Springer, Heidelberg (1982)
6. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement for symbolic model checking. J. ACM 50(5) (2003)
7. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
8. Graf, S., Saidi, H.: Construction of abstract state graphs with PVS. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254. Springer, Heidelberg (1997)
9. Holzmann, G.: The SPIN Model Checker. Addison Wesley, Reading (2004)
10. McMillan, K.: Symbolic Model Checking: An approach to the State Explosion Problem. Kluwer, Dordrecht (1993)
11. McMillan, K.: Interpolation and SAT-based Model Checking. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725. Springer, Heidelberg (2003)
12. Pnueli, A.: The temporal logic of programs. In: Proc. IEEE Symp. on Found. of Computer Science (1977)
13. Prasad, M., Biere, A., Gupta, A.: A survey on recent advances in SAT-based formal verification. Software Tools for Technology Transfer (STTT) 7(2) (2005)
14. Quielle, J., Sifakis, J.: Specification and verification of concurrent systems in CESAR. In: Dezani-Ciancaglini, M., Montanari, U. (eds.) Programming 1982. LNCS, vol. 137, Springer, Heidelberg (1982)
15. Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a SAT-solver. In: Johnson, S.D., Hunt Jr., W.A. (eds.) FMCAD 2000. LNCS, vol. 1954. Springer, Heidelberg (2000)
16. Veith, H., Grumberg, O. (eds.): 25 Years of Model Checking. LNCS, vol. 5000. Springer, Heidelberg (to be published, 2008)

Differential Algebra and System Modeling in Cellular Biology

François Boulier and François Lemaire

University Lille I, LIFL, 59655 Villeneuve d'Ascq, France
{Francois.Boulier,Francois.Lemaire}@lifl.fr
http://www.lifl.fr/~{boulier,lemaire}

Abstract. Among all the modeling approaches dedicated to cellular biology, differential algebra is particularly related to the well-established one based on nonlinear differential equations. In this paper, it is shown that differential algebra makes one of the model reduction methods both simple and algorithmic: the quasi-steady state approximation theory, in the particular setting of generalized chemical reactions systems. This recent breakthrough may suggest some evolution of modeling techniques based on nonlinear differential equations, by incorporating the reduction hypotheses in the models. Potential improvements of parameters fitting methods are discussed too.

Keywords: Computer algebra, differential algebra, cellular biology, system modeling.

1 Introduction

Among all the modeling approaches dedicated to cellular biology, differential algebra is particularly related to the well-established one based on nonlinear differential equations [1,2,3].

Differential equations apply to rather small models. There are many subapproaches, restricted to (sometimes piecewise) linear differential equations, qualitative simulations, ... [4,5,6]. The approach based on nonlinear differential equations was, however, successfully applied for the analysis of many genetic regulatory circuits, including those that work during complex animal embryogenesis [7]. One of the major successes of this approach is certainly the modeling of the segment polarity network of the drosophila [8,9].

This paper is mostly concerned with the model reduction problem for chemical reactions systems based on the generalized mass action law [10]. Among all the approximation techniques available for such systems [11, lumping, sensitivity analysis, ...], this paper is exclusively concerned with the quasi-steady state approximation theory.

The model reduction problem for systems of differential equations broadly consists in simplifying the given system, by means of some simplification hypotheses. It aims to get a tractable reduced system, while preserving some properties of interest (e.g. presence of oscillations, number of equilibria).

Modeling approaches based on nonlinear differential equations do not all directly rely on the formalism of chemical reactions systems. Quite often, more

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 22–39, 2008.

sophisticated functions are used such as the Henri-Michaelis-Menten factors and the Hill functions. It is worth noticing that these sophisticated functions can quite often be deduced from generalized chemical reactions systems through some model reduction. However, practitioners using these functions do not always explicitly formulate the simplifying assumptions which justify these approximations. Thereby, they do not formulate the domains of validity of their models as precisely as they could.

Differential algebra makes the quasi-steady state approximation method both simple and algorithmic in the particular setting of systems of generalized chemical reaction systems [12]. This fact thus suggests to widen the use of these systems for systems modeling, and to incorporate in the models the hypotheses which lead to the more sophisticated formulas. One should thereby expect to obtain models with more ascertained domains of validity and to estimate the practical relevance of some of the simplification hypotheses.

In addition to the above issues, the differential algebra and the quasi-steady state approximation theory also suggest improvements of the parameters fitting problem, which is an important concern in systems modeling. It has been known for a few years that differential elimination somehow transforms nonlinear least squares into linear ones in the parameters estimation problem, by suggesting a starting point for the Newton-like estimation methods [13,14]. By incorporating moreover, in these methods, the recent progress obtained in the quasi-steady state approximation theory, one may also expect to reduce the stiffness of the differential systems that they need to numerically integrate, thereby speeding up the overall optimization processes.

2 Differential Elimination

The content of this section owes a lot to [15]. We refer to that paper for more details.

The differential elimination theory is a subtheory of the differential algebra [16,17]. See also [18]. The differential elimination processes that are presented in this paper take as input two parameters: a system of polynomial (thus nonlinear) differential equations, ordinary or with partial derivatives[1] and a *ranking*[2]. They produce on the output an equivalent finite set of polynomial differential systems, which are simpler, in the sense that they involve some differential equations which are consequences of the input system but were somehow hidden. The output may consist of more than one differential system because the differential elimination process may need to split cases. The set of the differential equations which are consequences of the input system forms a so-called *differential ideal* of some *polynomial differential ring*. Since this ideal is an infinite set, a natural question arises: how does the process select the finitely many differential equations which appear in the output system? This is indeed the role of the rankings.

[1] This paper is only concerned with the ordinary case.

[2] Emphasized words have a technical meaning, which will be defined in section 2.2.

2.1 Example

The following example, borrowed from [19, Chapter VII, page 454], is a so-called differential algebraic equations system, since it mixes differential and non differential equations. There are three unknown time varying functions (three dependent variables) x, y and z. The dot over a variable denotes its derivative w.r.t. the independent variable, which is assumed to be the time, throughout this paper:

$$\dot{x} = 0.7\,y + \sin(2.5\,z), \quad \dot{y} = 1.4\,x + \cos(2.5\,z), \quad 1 = x^2 + y^2. \tag{1}$$

Let us try to apply an explicit numerical integration scheme such as Euler's one over this system, for some initial conditions. For each dependent variable (say) x, the scheme transforms the explicit differential equation $\dot{x} = f(x, y, z)$ as a recurrence formula $x_{n+1} = x_n + h\,f(x_n, y_n, z_n)$, where h denotes some stepsize, and computes the terms x_n the ones after the others. However, the absence of a differential equation of the following form seems to forbid the application of Euler's scheme:

$$\dot{z} = f(x, y, z) \tag{2}$$

Indeed, this equation is not missing. It is a "hidden" consequence of the input system: it can be automatically extracted from it by differential elimination. In order to show how to proceed with the help of the *diffalg* package of MAPLE, one first converts the system as a *polynomial differential system*. For this, one denotes s the sine, c the cosine and one introduces a few more equations. The following differential polynomial system is equivalent to the above one.

$$\begin{aligned} \dot{x} = 0.7\,y + s, \quad \dot{y} = 1.4\,x + c, \quad 1 = x^2 + y^2, \\ \dot{s} = 2.5\,\dot{z}\,c, \quad \dot{c} = -2.5\,\dot{z}\,s, \quad 1 = s^2 + c^2. \end{aligned} \tag{3}$$

In order to compute equation (2) using *diffalg*, one stores the differential polynomial system in the variable *syst*, converting floating point numbers as rational numbers.

```
with (diffalg):
syst := [diff(x(t),t) - 7/10*y(t) - s(t),
         diff(y(t),t) - 14/10*x(t) - c(t),
         x(t)^2 + y(t)^2 - 1,
         diff(s(t),t) - 25/10*diff(z(t),t)*c(t),
         diff(c(t),t) + 25/10*diff(z(t),t)*s(t),
         s(t)^2 + c(t)^2 - 1]:
```

Then the variable R is assigned the context of the computation: the only derivation is taken with respect to the time, the notation is the standard *diff* notation of MAPLE and the *ranking* is provided. According to the ranking notation of the *diffalg* package, this is the *orderly* ranking such that $s > c > x > y > z$. The properties of that ranking ensure that, if there exists a differential equation of the form (2) in the *radical* of the *differential ideal* generated by *syst* then the differential elimination process will find it.

```
R := differential_ring (derivations = [t], notation = diff,
                        ranking = [[s, c, x, y, z]]):
```

Next the *Rosenfeld-Gröbner* function [20] is applied to *syst* and R. It returns a list of MAPLE tables. Each table provides a *regular differential chain* defining some differential ideal. The list should be understood as an intersection. Over the example, the list only involves one entry so that the regular differential chain does represent the radical differential ideal generated by the input system. The desired equation stands on the second place of the chain (only the two first equations are displayed). Enlarging the input system with this equation, it is now easy to perform any numerical integration method and our problem is solved.

```
ideal := Rosenfeld_Groebner (syst, R):
rewrite_rules (ideal [1]);
```

$$\Big[\frac{d}{dt} y(t) = \frac{7}{5} x(t) + c(t),$$

$$\frac{d}{dt} z(t) = \frac{1}{25} \frac{3500 - 12348\ (y(t))^6 + 13230\, c(t)\, x(t)\, (y(t))^4 + 25809\ (y(t))^4}{441\ (y(t))^6 - 882\ (y(t))^4 + 541\ (y(t))^2 - 100}$$

$$+ \frac{1}{25} \frac{-14700\, x(t)\, (y(t))^2\, c(t) - 16961\ (y(t))^2 + 3940\, x(t)\, c(t)}{441\ (y(t))^6 - 882\ (y(t))^4 + 541\ (y(t))^2 - 100}, \quad \dots \Big]$$

2.2 Differential Algebra

A *differential ring* (resp. field) is a ring (resp. field) R endowed with a derivation (this paper is restricted to the case of a single derivation but the theory is more general) i.e. a unitary mapping $R \to R$ such that (denoting $\dot{a}$ the derivative of a):

$$\dot{(a+b)} = \dot{a} + \dot{b}, \quad \dot{(a\,b)} = \dot{a}\,b + a\,\dot{b}. \tag{4}$$

Observe that, theoretically, the derivation is an abstract operation. For legibility, one views it as the derivation w.r.t. the time t. Algorithmically, one is led to manipulate finite subsets of some *differential polynomial ring* $R = K\{U\}$ where K is the differential field of coefficients (in practice, $K = \mathbb{Q}$, $\mathbb{Q}(t)$ or $\mathbb{Q}(k_1, \dots, k_r)$ where the k_i denote parameters that would be assumed to be algebraically independent) and U is a finite set of *dependent variables*[3]. The elements of R, the *differential polynomials* are just polynomials in the usual sense, built over the infinite set, denoted ΘU, of all the derivatives of the dependent variables.

[3] In the differential algebra theory, the terminology *differential indeterminates* is preferred to *dependent variables* for derivations are abstract and differential indeterminates are not even assumed to correspond to functions. In order not to mix different expressions in this paper, the second expression, which seems to be more widely known, was chosen.

Definition 1. *A* differential ideal *of a differential ring R is an ideal of R, stable under the action of the derivation.*

Let F be a finite subset of a differential ring R. The set of all the finite linear combinations of various orders derivatives of elements of F, with elements of R for coefficients, is a differential ideal. It is called the differential ideal generated by F. An ideal $\mathfrak{A}$ is said to be *radical* if $a \in \mathfrak{A}$ whenever there exists some nonnegative integer p such that $a^p \in \mathfrak{A}$. The radical of an ideal $\mathfrak{A}$ is the set of all the ring elements a power of which belongs to $\mathfrak{A}$. The radical of a (differential) ideal is a radical (differential) ideal.

Theorem 1. *Let R be a differential polynomial ring and F be a finite subset of R. A differential polynomial p of R lies in the radical of the differential ideal generated by F if and only if it vanishes over every analytic solution of F.*

Proof. [16, chap. II, §7, 11] or [21].

The *Rosenfeld-Gröbner* algorithm [20] solves the membership problem to radical differential ideals. To present it, one needs to define the concept of *ranking*.

Definition 2. *If U is a finite set of dependent variables, a* ranking *over U is a total ordering over the set ΘU of all the derivatives of the elements of U which satisfies: $a < \dot{a}$ and $a < b \Rightarrow \dot{a} < \dot{b}$ for all a, $b \in \Theta U$.*

Let U be a finite set of dependent variables. A ranking such that, for every u, $v \in U$, the ith derivative of u is greater than the jth derivative of v whenever $i > j$ is said to be *orderly*. If U and V are two finite sets of dependent variables, one denotes $U \gg V$ every ranking such that any derivative of any element of U is greater than any derivative of any element of V. Such rankings are said to *eliminate* U w.r.t. V.

Assume that some ranking is fixed. Then one may associate with any differential polynomial $f \in K\{U\} \setminus K$ the greatest (w.r.t. the given ranking) derivative $v \in \Theta U$ such that $\deg(f, v) > 0$. This derivative is called the *leading derivative* or the *leader* of f.

Rankings permit to define leaders. Leaders permit to use differential polynomial as rewrite (substitution) rules. Assume that $f = a_d\, v^d + \cdots + a_1\, v + a_0$ is a differential polynomial with leader v (the coefficients a_i are themselves differential polynomials). Then the equation $f = 0$ can be written (as the *rewrite_rules* function of *diffalg* presented in section 2.1 does, though it uses an equal sign instead of an arrow):

$$v^d \longrightarrow -\frac{a_{d-1}\, v^{d-1} + \cdots + a_1\, v + a_0}{a_d}\,. \tag{5}$$

It can be used afterwards as a rule to simplify any differential polynomial g such that $\deg(g, v) \geq d$ or $\deg(g, v^{(k)}) > 0$ where $v^{(k)}$ denotes any proper derivative of v. There are precise algorithms for performing these sorts of substitution by finite sets of rewrite rules: Ritt's reduction algorithm or the *normal form* algorithm [21, algorithm NF].

The *Rosenfeld-Gröbner* algorithm gathers as input a finite system F of differential polynomials and a ranking. It returns a finite family (possibly empty) $C_1, \ldots, C_r$ of finite subsets of $K\{U\} \setminus K$, called *regular differential chains*. Each system C_i defines a differential ideal $\mathfrak{C}_i$ (it is a *characteristic set* of $\mathfrak{C}_i$) in the sense that, for any $f \in K\{U\}$, we have

$$f \in \mathfrak{C}_i \quad \text{iff} \quad \mathrm{NF}(f, C_i) = 0. \tag{6}$$

The relationship with the radical $\mathfrak{A}$ of the differential ideal generated by F is the following:

$$\mathfrak{A} = \mathfrak{C}_1 \cap \cdots \cap \mathfrak{C}_r. \tag{7}$$

When $r = 0$ we have $\mathfrak{A} = K\{U\}$. Combining both relations, one gets an algorithm to decide membership in $\mathfrak{A}$. Indeed, given any $f \in K\{U\}$ we have:

$$f \in \mathfrak{A} \quad \text{iff} \quad \mathrm{NF}(f, C_i) = 0, \quad 1 \leq i \leq r. \tag{8}$$

The differential ideals $\mathfrak{C}_i$ do not need to be prime. They are however necessarily radical. The $\mathrm{NF}(\,\cdot, yC_i)$ function permits to compute canonical representatives of the residue classes of the differential ring $R/\mathfrak{C}_i$.

3 Quasi-Steady State Approximation for Generalized Chemical Reactions Systems

Differential elimination makes the quasi-steady state approximation (QSSA) theory both simple and algorithmic in the special setting of generalized[4] chemical reactions systems as shown by [12].

3.1 QSSA in General: Fast and Slow Variables

In principle, QSSA applies to systems under the two time-scales standard form. Consider the following system in two dependent variables x and y (for legibility, but there may be more than two variables) and assume that ε is a small positive number.

$$\dot{x} = f(x, y), \quad \varepsilon\, \dot{y} = g(x, y). \tag{9}$$

On a random point $(x, y) \in \mathbb{R}^2$, the speed of y is high and thus, under some general conditions, rapidly approaches an area where $g(x, y) \simeq 0$. The variable x is said to be *slow* while the variable y is said to be fast. The QSSA amounts to approximate such a system by the following one:

$$\dot{x} = f(x, y), \quad 0 = g(x, y). \tag{10}$$

which mixes differential and algebraic equations. Note that this approximation is only valid under some conditions (e.g. stability of the differential system in

[4] Chemical reactions systems are said to be *generalized* when their elementary reactions are not required to be balanced. See [10].

the neighborhood of $g(x, y) = 0$) given in the Tikhonov theorem [22] and after the transient step is elapsed.

Performing a QSSA over a differential system presents two advantages: it reduces the number of ODE and it tends to transform stiff systems into nonstiff ones, much easier to solve numerically.

However, the QSSA is not proven algorithmic in general. The issue relies in the fact that the fast and slow variables, if they exist, may only be obtained through a change of coordinates and there does not seem to exist any algorithm which decides, given a differential system, if such a change of coordinates does exist.

3.2 QSSA for Chemical Reactions Systems: Fast and Slow Reactions

For differential systems arising from generalized chemical reactions systems, there exists a standard way to perform the QSSA, provided that the set of chemical reactions is divided in two parts: the fast ones and the slow ones.

As far as we know, the first clear relationship between this method and the Tikhonov theorem was established in [23]. Afterwards, close variants of the same method were rediscovered more or less independently [24,25]. Though all these papers present methods, none of them is fully presented in an algorithmic manner. This may at least partly be due to the fact that some steps of the methods require the inversion of a matrix over a residue class ring, a non obvious task which may imply splitting cases.

Indeed, it turns out that the whole method is equivalent to a differential elimination process, as shown for the first time in [12]. For a general presentation of the method, one refers to [12]. In this paper, one only presents the method over a famous example: the Henri-Michaelis-Menten reduction of the basic enzymatic reaction system:

$$E + S \underset{k_2}{\overset{k_1}{\rightleftarrows}} C \xrightarrow{k_3} E + P. \tag{11}$$

The initial system of ODE writes: $\dot{X} = N\,V$ i.e.

$$\begin{pmatrix} \dot{E} \\ \dot{C} \\ \dot{S} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} k_1\,E\,S \\ k_2\,C \\ k_3\,C \end{pmatrix} \tag{12}$$

where X is the vector of the chemical species, N is the system stoichiometry matrix and V is the vector of the reaction rates. The stoichiometry matrix is built as follows: it involves one row per species and one column per reaction. The entry at row r, column c is the number of molecules of species r produced by the reaction c (i.e. the number of times species r occurs on the reaction right handside minus the number of times it occurs on the reaction left handside). The rate of a reaction is the product of the left handside species (with multiplicities)

times the reaction rate (the parameter over the arrow). Expanding the formula, one gets:

$$\begin{aligned} \dot{E} &= -k_1\, E\, S + (k_2 + k_3)\, C, \\ \dot{S} &= -k_1\, E\, S + k_2\, C, \\ \dot{C} &= k_1\, E\, S - (k_2 + k_3)\, C, \\ \dot{P} &= k_3\, C. \end{aligned} \tag{13}$$

Among all the assumptions leading to the Henri-Michaelis-Menten formula:

$$\dot{S} = -\frac{V_{\max}\, S}{K + S}, \tag{14}$$

the only one concerning the QSSA is that k_1, $k_2 \gg k_3$ i.e. that the two leftmost reactions of the chemical system (11) are fast while the rightmost one is slow. The other assumptions will be given later.

In order to perform the QSSA over the above system, one builds a DAE system from system (13) by replacing the contribution of the fast reactions by a new dependent variable, F_1, and by inserting an algebraic equation, defining the "slow variety", stating that the fast reactions are at quasi-equilibrium.

$$\begin{aligned} \dot{E} &= -F_1 + k_3\, C, \\ \dot{S} &= -F_1, \\ \dot{C} &= F_1 - k_3\, C, \\ \dot{P} &= k_3\, C, \\ 0 &= k_1\, E\, S - k_2\, C. \end{aligned} \tag{15}$$

In general there may be many different new dependent variables F_i and many different algebraic equations. Observe that, over the example, the two fast reactions are considered as one reversible reaction and associated to only one dependent variable: F_1. The general process is described in section 3.3. Differential elimination is performed below using *diffalg*. The parameters are put in the base field $\mathscr{F}$ of the equations to avoid discussions over their values (they are then considered as algebraically independent). The inequation $C(t) \neq 0$ is inserted to avoid considering this degenerate case. The output is pretty printed.

```
with(diffalg):
DAE_syst := [
    diff(E(t),t) - (-F1(t) + k3*C(t)),
    diff(S(t),t) + F1(t),
    diff(C(t),t) - (F1(t) - k3*C(t)),
    diff(P(t),t) - k3*C(t),
    k1*E(t)*S(t) - k2*C(t) ]:
F := field_extension (transcendental_elements=[k1,k2,k3]):
R := differential_ring (ranking=[[F1], [C,E,P,S]], notation=diff,
                        derivations=[t], field_of_constants=F):
ideal := Rosenfeld_Groebner (DAE_syst, [C(t)], R);
                        ideal := [characterizable]
rules := rewrite_rules(ideal [1]);
```

$$\left[F_1 = \frac{k_3\, k_1\, E\, S\, (k_1\, S + k_2)}{k_2\ (k_1\, S + k_1\, E + k_2)}, \quad \dot{E} = \frac{k_1^2\, E^2\, k_3\, S}{k_2\ (k_1\, S + k_1\, E + k_2)}, \quad \dot{P} = \frac{k_3\, k_1\, E\, S}{k_2},\right.$$

$$\left.\dot{S} = -\frac{k_3\, k_1\, E\, S\, (k_1\, S + k_2)}{k_2\ (k_1\, S + k_1\, E + k_2)}, \quad C = \frac{k_1\, E\, S}{k_2}\right]$$

Of course, one does not recognize formula (14) in this regular differential chain: the reduction is incomplete since the extra assumptions made in the Henri-Michaelis-Menten reduction have not yet been taken into account. These assumptions are: $S(0) \gg E(0)$, $P \simeq 0$ and $C(0) = 0$. One refers to [12] for a complete reduction.

3.3 Construction of the DAE to Consider for the QSSA

First split the stoichiometry matrix N into two matrices $N\mathrm{f}$ and N_s putting the columns which correspond to fast reactions in N_f and the ones which correspond to slow reactions in N_s. Split accordingly the rows of the vector V into two vectors V_f and V_s. One gets a formula $\dot{X} = N_\mathrm{s}\, V_\mathrm{s} + N_\mathrm{f}\, V_\mathrm{f}$. Over system (12), one gets:

$$\begin{pmatrix} \dot{E} \\ \dot{C} \\ \dot{S} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 0 \\ 1 \end{pmatrix} \cdot \left(k_3\, C \right) + \begin{pmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} k_1\, E\, S \\ k_2\, C \end{pmatrix}. \tag{16}$$

Determine a maximal linearly independent set of columns of N_f (i.e. a basis of that matrix) and remove the other ones, giving a new matrix $\overline{N}_\mathrm{f}$. Update the vector of reaction rates V_f, giving a new vector $\overline{V}_\mathrm{f}$ such that $N_\mathrm{f}\, V_\mathrm{f} = \overline{N}_\mathrm{f}\, \overline{V}_\mathrm{f}$. Over the example, removing the second column, one gets a new formula $\dot{X} = N_\mathrm{s}\, V_\mathrm{s} + \overline{N}_\mathrm{f}\, \overline{V}_\mathrm{f}$ giving system (13):

$$\begin{pmatrix} \dot{E} \\ \dot{C} \\ \dot{S} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 0 \\ 1 \end{pmatrix} \cdot \left(k_3\, C \right) + \begin{pmatrix} -1 \\ 1 \\ -1 \\ 0 \end{pmatrix} \cdot \left(k_1\, E\, S - k_2\, C \right). \tag{17}$$

Replace the vector $\overline{V}_\mathrm{f}$ by a vector F of new dependent variables F_i. The slow variety is defined by letting the entries of $\overline{V}_\mathrm{f}$ all equal to zero. The DAE to be considered for quasi-steady state approximation is

$$\dot{X} = N_\mathrm{s}\, V_\mathrm{s} + \overline{N}_\mathrm{f}\, F, \qquad \overline{V}_\mathrm{f} = 0. \tag{18}$$

Over the example, one gets a formula giving system (15):

$$\begin{pmatrix} \dot{E} \\ \dot{C} \\ \dot{S} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 0 \\ 1 \end{pmatrix} \cdot \left(k_3\, C \right) + \begin{pmatrix} -1 \\ 1 \\ -1 \\ 0 \end{pmatrix} \cdot \left(F_1 \right), \qquad \left(k_1\, E\, S - k_2\, C \right) = 0. \tag{19}$$

3.4 Limits and Generalizations of the Method

In some cases, the F_i variables cannot be all eliminated. In that case, the method does not apply. In [12], this is checked by testing whether $\mathrm{NF}(\dot{X}_i, C)$ only depends on the X_j (with order 0), for each dependent variable X_i and each regular differential chain C produced by differential elimination. A simple example is given by the generalized chemical reactions system, assuming k_1, k_2 are both fast:

$$A \xrightarrow{k_1} B, \quad A \xrightarrow{k_2} C. \tag{20}$$

If one applies the method sketched above, one gets the following differential-algebraic system:

$$\dot{A} = -F_1 - F_2, \quad \dot{B} = F_1, \quad \dot{C} = F_2, \quad 0 = k_1\, A, \quad 0 = k_2\, A \tag{21}$$

which simplifies to the regular differential chain:

$$F_1 = -\dot{C}, \quad F_2 = \dot{C}, \quad \dot{B} = -\dot{C}, \quad A = 0. \tag{22}$$

The normal form of $\dot{B}$ is $-\dot{C}$ and $\dot{C}$ is equal to its own normal form: the method failed.

According to the Tikhonov theorem, there are some extra conditions to check for the approximation to be valid [23, conditions C3 and C4]. In particular, the slow variety defined by the algebraic equations must be attractive.

In some cases, there exists a better slow variety than the one provided by the method. To compute a reduced system w.r.t. this different variety, one just needs to change the set of algebraic equations of the differential-algebraic system and to run the differential elimination process. Such a situation is illustrated by the next example borrowed from [26] where $k_2 \gg k_1$:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} \varnothing. \tag{23}$$

If one applies the method, one gets the following differential-algebraic system:

$$\dot{A} = -k_1\, A, \quad \dot{B} = k_1\, A - F_1, \quad 0 = k_2\, B \tag{24}$$

which simplifies to the following regular differential chain:

$$F_1 = k_1\, A, \quad \dot{A} = -k_1\, A, \quad B = 0. \tag{25}$$

However, the slow variety $k_1\, A - k_2\, B = 0$ is better than $B = 0$. A better reduced system is thus obtained by performing differential elimination over the following differential-algebraic system:

$$\dot{A} = -k_1\, A, \quad \dot{B} = k_1\, A - F_1, \quad 0 = k_1\, A - k_2\, B \tag{26}$$

which simplifies to the following regular differential chain:

$$F_1 = (k_1 + k_2)\, B, \quad \dot{B} = -k_1\, B, \quad A = \tfrac{k_2}{k_1}\, B. \tag{27}$$

4 Application to System Modeling in Cellular Biology

This section aims at establishing some relationship between the quasi-steady state approximation method sketched in section 3 and system modeling in cellular biology. For this purpose, an application of this technique to the analysis of a genetic regulatory circuit made of a single gene, regulated by a polymer of its own protein [27,28] is described.

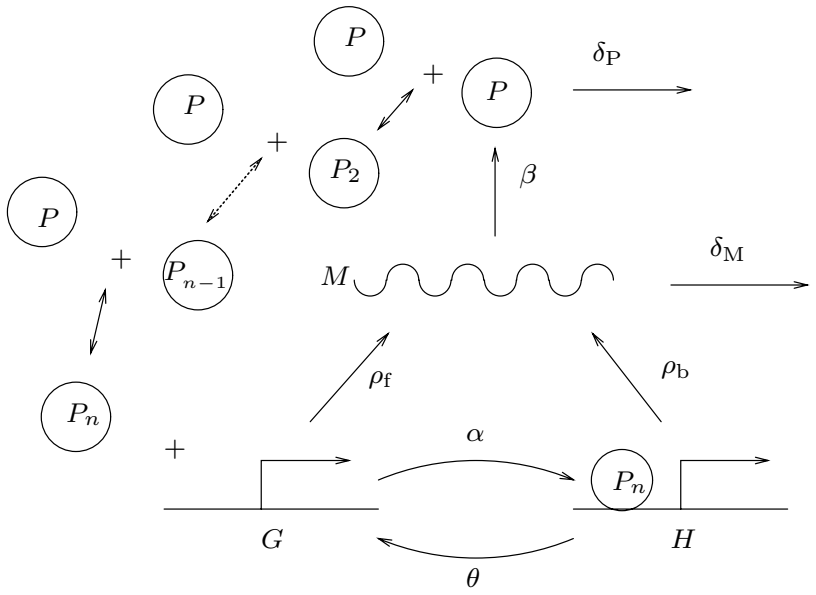


Fig. 1. A single gene regulated by a polymer of its own protein

One considers the genetic circuit depicted in Figure 1. The single gene is regulated by an order n polymer of its own protein. The integer number n is a parameter of the system. This study was motivated by the activity of a working group aiming at modeling the circadian clock of the green alga *ostreococcus tauri*. The addressed question was: does there exist biologically meaningful parameters values which make this circuit oscillate? More technically: does there exist biologically meaningful parameters values which make a Poincaré-Andronov-Hopf bifurcation occur? One refers to [27] for a more detailed motivation of the addressed question and to [29] for a related work.

There are many different ways to derive a system of ordinary differential equations from the considered circuit but one of the simplest schemes consists in first translating it as a system of generalized chemical reactions (observe that transcription and translation are not balanced reactions). The variables G and H represent the state of the gene. The mRNA concentration and the concentration of the protein translated from the mRNA are represented by M and P. The n types of polymers of P are denoted by $P = P_1, P_2, \ldots, P_n$. Greek letters and k_i^-, k_i^+ $(1 \leq i \leq n-1)$ represent parameters:

$$\begin{gathered} G + P_n \underset{\theta}{\overset{\alpha}{\rightleftarrows}} H, \quad G \xrightarrow{\rho_{\mathrm{f}}} G + M, \quad H \xrightarrow{\rho_{\mathrm{b}}} H + M, \\ M \xrightarrow{\beta} M + P, \quad M \xrightarrow{\delta_{\mathrm{M}}} \varnothing, \quad P \xrightarrow{\delta_{\mathrm{P}}} \varnothing, \\ P_i + P \underset{k_i^-}{\overset{k_i^+}{\rightleftarrows}} P_{i+1} \quad (1 \leq i \leq n-1). \end{gathered} \tag{28}$$

This generalized chemical reactions system can now be canonically translated as a system of parametric ordinary differential equations, denoting $A_i = (k_i^- P_{i+1} - k_i^+ P_i P)$. Variables G, H, M, $P = P_1, \ldots, P_n$ are dependent variables. They all represent species concentrations except G and H, which should rather be viewed as "random variables".

$$\begin{aligned} \dot{G} &= \theta H - \alpha G P_n, \\ \dot{H} &= -\theta H + \alpha G P_n, \\ \dot{M} &= \rho_{\mathrm{f}} G + \rho_{\mathrm{b}} H - \delta_{\mathrm{M}} M, \\ \dot{P} &= \beta M - \delta_{\mathrm{P}} P + 2 A_1 + A_2 + \cdots + A_{n-1}, \\ \dot{P}_i &= -A_{i-1} + A_i \qquad (2 \leq i \leq n-1), \\ \dot{P}_n &= -A_{n-1} + \theta H - \alpha G P_n. \end{aligned} \tag{29}$$

This system of $n+3$ differential equations depending on $2n+5$ parameters is actually much too large for any further symbolic analysis. It needs to be reduced.

In order to apply a quasi-steady state approximation, it is assumed that the $n-1$ chemical reactions describing the polymerization of the protein are fast compared to the other ones.

Then, according to the technique sketched in section 3, one gets an approximation of system (29) by replacing each expression A_i by a new dependent variable F_i $(1 \leq i \leq n-1)$ and by augmenting this system by the $n-1$ following algebraic equations:

$$0 = k_i^+ P P_i - k_i^- P_{i+1}, \qquad (1 \leq i \leq n-1). \tag{30}$$

It is now sufficient to eliminate the F_i from the so obtained differential-algebraic system. Unfortunately, this cannot be performed by a standard differential elimination algorithm since the number of equations depends on the parameter n. However, such an algorithm can be applied for many different values of n and the general formula can be inferred.

$$\begin{aligned} \dot{G} &= \theta H - \alpha K_{n-1} P^n G, \\ \dot{H} &= -\theta H + \alpha K_{n-1} P^n G, \\ \dot{M} &= \rho_{\mathrm{b}} H + \rho_{\mathrm{f}} G - \delta_{\mathrm{M}} M, \\ \dot{P} &= \frac{n \theta H - n \alpha K_{n-1} P^n G - \delta_{\mathrm{P}} P + \beta M}{\displaystyle\sum_{i=0}^{n-1} (i+1)^2 K_i P^i} \end{aligned} \tag{31}$$

where $K_i = \dfrac{k_1^+ \cdots k_i^+}{k_1^- \cdots k_i^-}$ with the convention $K_0 = 1$.

The redundant equation describing the evolution of H can be removed and H can be replaced by $\gamma_0 - G$ in the three remaining equations, where γ_0 is some new parameter denoting the total quantity of gene. Some further exact reduction of the parameters set can moreover be performed and one is led to the three parametric ODE model [28, system (3)].

This three parametric ODE reduced system is simple enough to be tackled by the Hopf criterion (a variant of the Routh-Hurwitz one) and it was proven in [27,28] that a Poincaré-Andronov-Hopf bifurcation occurs for biologically meaningful values of the parameters if and only if $n > 8$. One refers to those papers for a biological interpretation of this result.

Let us conclude this section by a few comments on the involved quasi-steady state approximation methods. When [27] was written, its authors did not know the particular way to perform the quasi-steady state approximation over an ODE system derived from a generalized chemical reactions system. Thus they applied a "handmade" technique inspired from the singular perturbation theory yielding a three ODE model and "solved" it. Integral curves obtained by numerical integration showed differences between the initial and the reduced system but this was expected. Later, the same authors understood the particular technique for generalized chemical reactions system and showed how it is related to differential elimination [12]. This technique, applied over the very same system in [28], produced (after a suitable change of coordinates) a slightly different reduced system: one gets the old [27] reduced system by clearing the denominator of the last ODE of system (31). By some reduction argument, the proof of the old paper could be applied to the new model. This somehow confirmed that the handmade reduction technique was good. However, comparisons of integral curves obtained by numerical integration showed that the new model is more accurate, in the sense that it has a wider domain of validity, than the old one.

5 Parameters Estimation

The problem which is addressed in this section can be stated as follows: given a parametric ordinary differential system and files of measures for some of the dependent variables, estimate the values of the unknown parameters. In this context, differential elimination somehow transforms a nonlinear least squares problem into a linear one by guessing a starting point for a Newton-like method.

The approach described in this section was published in [30,13]. It was implemented in a software based on the BLAD libraries [31,14]. It is strongly related to the study of the identifiability of parametric differential systems, for which a huge literature is available [32,33,34,35,36].

To be honest, the method is not very convenient for system modeling in cellular biology for it requires pretty precise model equations as well as pretty accurate measures. In cellular biology, one usually does not know the model equations though the situation might be better for in vitro systems or synthetic genes. But accurate measures are definitely not available in 2008. However, we chose to present the method anyway in this paper for two reasons: first, the use

of differential elimination always improves the classical numerical one; second, the quasi-steady state approximation technique presented in section 3 suggests another, new, improvement.

The content of this section owes a lot to [15] and one refers to this paper for more detailed explanations.

The problem is stated by an academic example. Consider the following system of parametric ordinary differential equations. There are two dependent variables x_1, x_2 and four parameters k_{12}, k_{21}, k_e, V_e. It could be derived from a chemical reactions system since it involves two linear exchanges and a Henri-Michaelis-Menten degradation term.

$$\dot{x}_1 = -k_{12}\, x_1 + k_{21}\, x_2 - \frac{V_e\, x_1}{k_e + x_1}, \quad \dot{x}_2 = k_{12}\, x_1 - k_{21}\, x_2. \tag{32}$$

Assume that x_1 is observed i.e. that a file of measures is available for this dependent variable but that x_2 is not. For the sake of simplicity, assume moreover that $x_2(0) = 0$ and $k_e = 7$ are known. Issue: estimate the values of the three unknown parameters k_{12}, k_{21}, V_e.

There exists a purely numerical method to solve this problem. It is based on a nonlinear least squares method i.e. a Newton method. The idea is simple: pick random values for the three unknown parameters. Integrate numerically the differential system w.r.t. these values and compare the curve obtained by simulation with the file of measures. The error is defined as the sum, for all abscissas, of the squares of the ordinates differences between the two curves. The Newton method (a Levenberg-Marquardt scheme was applied in [14]) updates the values of the three unknown parameters if the error is considered as too large. It stops either if the error is small enough of if a stationary point is reached. However, as every Newton method, nonlinear least squares are very sensitive to the choice of the starting point (the initial random values) and are very likely to end up in a local minimum, with wrong parameters values.

By means of differential elimination, it is sometimes possible to compute a first estimate of the unknown parameters. This estimate is usually not very precise but can be used as a starting point for the Newton method.

The idea consists in eliminating the non observed variables in order to get a relation which only involves the observed variable x_1, its derivatives up to any order and the parameters. Let us proceed with the help of *diffalg*. The *Rosenfeld-Gröbner* algorithm[5] is applied over the model equations. The ranking $x_2 \gg x_1$ eliminates x_2 w.r.t. x_1. The right-hand side of the first model equation is a rational fraction. It is decomposed as a numerator and a denominator. The numerator is stored in the list of the equations (first parameter to *Rosenfeld-Gröbner*). The denominator is stored in the list of the *inequations*. To avoid splitting cases on parameters values, one views them as algebraically independent elements of the base field of the differential polynomials.

[5] There are more efficient algorithms than *Rosenfeld-Gröbner* for performing this elimination since the input system already is a characteristic set w.r.t. some orderly ranking: one could apply a change of ranking algorithm [33,37] which would avoid splitting cases.

```
eq1 := diff (x1(t),t) + k12*x1(t) - k21*x2(t) + Ve*x1(t)/(ke + x1(t));
eq2 := diff (x2(t),t) - k12*x1(t) + k21*x2(t);
K := field_extension (transcendental_elements = [k21, k12, ke, Ve]):
R := differential_ring
             (derivations = [t], notation = diff,
              field_of_constants = K, ranking = [x2, x1]):
ideal := Rosenfeld_Groebner ([numer (eq1), eq2], [denom (eq1)], R);
```

$$ideal := [characterizable]$$

The characteristic set *ideal* involves two polynomials. The one which does not involve x_2 is the second one, which is displayed below, slightly pretty printed. The expressions enclosed between square brackets are called "parameters blocks".

$$\ddot{x}_1\,(x_1+k_\mathrm{e})^2+[k_{12}+k_{21}]\,\dot{x}_1\,(x_1+k_\mathrm{e})^2+[V_\mathrm{e}]\,\dot{x}_1\,k_\mathrm{e}+[k_{21}\,V_\mathrm{e}]\,x_1\,(x_1+k_\mathrm{e})=0.$$

This equation tells us that the systems parameters are in principle uniquely defined. Indeed, assume that the function x_1 is known. Then so are its derivatives $\dot{x}_1$ and $\ddot{x}_1$. These three functions can therefore be evaluated for three different values of the time t. The known parameter k_e can be replaced by its value. One thereby gets an exactly determined system of three linear equations whose unknowns are the parameters blocks. This system admits a unique solution. The values of the parameters blocks being fixed, it is obvious (in this example!) that the values of k_{12}, k_{21} and V_e also are uniquely defined. QED.

In practice, the function x_1 is known from a file of measures and one can try to numerically estimate the values of its first and its second derivative. If the measures are free of noise, the first derivative can be quite accurately estimated but this is usually not the case for the second derivative. To overcome these difficulties due to numerical approximations, one builds an overdetermined linear system that one solves by means of linear least squares.

The values of the blocks of parameters being known, one still has to recover the values of the parameters by solving the above algebraic system. In this example, it is very easy. The obtained values can then be used as a starting point for the Newton method. Observe that one cannot guarantee that this starting point does actually lead to the global minimum.

There are however two more important difficulties to overcome.

There may exist algebraic relations between the parameters blocks. There is no such relation in the example. But assume, for the sake of the explanation, that the computed differential polynomial involves the three following blocks of parameters so that the third block is the product of the two first ones:

$$[V_\mathrm{e}],\quad [k_{21}],\quad [V_\mathrm{e}\,k_{21}]. \tag{33}$$

There is no doubt that the numerical values produced during the resolution of the linear overdetermined system would not satisfy this relation. This would imply that the final algebraic system to solve in order to get the values of the parameters would be inconsistent.

The differential systems that the Newton method needs to numerically integrate at runtime may become stiff, forcing the numerical integrators to choose very small step sizes, thus slowing down the whole optimisation process. Indeed, in these nonlinear fitting methods, the parameters are the variables. The sets of large and small parameters have to change during the process and stiffness is often caused by the presence of different time scales in the differential systems.

A promising idea would consist, in the context of systems evolving from generalized chemical reactions systems, in applying the quasi-steady state approximation technique described in section 3. The differential system to be integrated could be replaced by a reduced one. The error computation of the Newton method could then be performed over the reduced system, speeding up the overall process.

6 Conclusion

As a conclusion, let us respond to a quotation of Eric Ponder (a director of Long Island Biology Association), borrowed from [38, Postscript]: "work on the mathematics [for biology] seems to me to have developed along two equally unprofitable lines".

The use of computer algebra methods for performing differential elimination seems very promising for enhancing (at least) software for system modeling in cellular biology. The recent progresses in the quasi-steady state approximation theory suggest to widen the use of generalized chemical reactions systems and make the simplification hypotheses more explicit which would otherwise justify the use of Henri-Michaelis-Menten or Hill terms. Already available standalone C libraries for differential elimination [31] should still simplify this evolution.

In turn, the search of applications in the field of system modeling in cellular biology also led to improvements in differential algebra. It is this work which pushed the authors to study the quasi-steady state approximation theory in the context of generalized chemical reactions systems and which has also suggested a new improvement of the nonlinear least squares methods.

Last, these works are far from over. For instance, an important theoretical challenge consists in designing an algorithmic method for determining good slow varieties of generalized chemical reactions systems. The authors are involved in another, more concrete challenge: to develop a modeling software based on these ideas and to use it to design and analyze synthetic gene networks.

References

1. Leloup, J.C., Goldbeter, A.: Modeling the molecular regulatory mechanism of circadian rhythms in Drosophila. Bioessays 22, 84–93 (2000)
2. Fall, C.P., Marland, E.S., Wagner, J.M., Tyson, J.J.: Computational Cell Biology. Interdisciplinary Applied Mathematics, vol. 20. Springer, Heidelberg (2002)
3. Conrad, E.D., Tyson, J.J.: Modeling Molecular Interaction Networks with Nonlinear Differential Equations. In: Szallasi, Z., Stelling, J., Periwal, V. (eds.) System Modeling in Cellular Biology: From Concepts to Nuts and Bolts, pp. 97–124. The MIT Press, Cambridge (2006)

4. de Jong, H.: Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. Journal of Computational Biology 9(1), 67–103 (2002)
5. de Jong, H., Geiselmann, J., Hernandez, C., Page, M.: Genetic Network Analyzer: qualitative simulation of genetic regulatory networks. Bioinformatics 19(3), 336–344 (2003)
6. de Jong, H., Ropers, D.: Qualitative Approaches to the Analysis of Genetic Regulatory Networks. In: Szallasi, Z., Stelling, J., Periwal, V. (eds.) System Modeling in Cellular Biology: From Concepts to Nuts and Bolts, pp. 125–147. The MIT Press, Cambridge (2006)
7. Saka, Y., Smith, J.C.: A mechanism for the sharp transition of morphogen gradient interpretation in Xenopus. BMC Dev. Biol. 7(47) (2007)
8. von Dassow, G., Meir, E., Munro, E.M., Odell, G.M.: The segment polarity network is a robust developmental module. Nature 406, 188–192 (2000)
9. von Dassow, G., Meir, E.: Exploring modularity with dynamical models of gene networks. In: Schlosser, G., Wagner, G.P. (eds.) Modularity in Development and Evolution, pp. 245–287. University of Chicago Press (2003)
10. Horn, F., Jackson, R.: General mass action kinetics. Archive for Rational Mechanics and Analysis 47, 81–116 (1972)
11. Okino, M.S., Mavrovouniotis, M.L.: Simplification of Mathematical Models of Chemical Reaction Systems. Chemical Reviews 98(2), 391–408 (1998)
12. Boulier, F., Lefranc, M., Lemaire, F., Morant, P.E.: Model Reduction of Chemical Reaction Systems using Elimination. In: The international conference MACIS 2007 (2007), http://hal.archives-ouvertes.fr/hal-00184558
13. Denis-Vidal, L., Joly-Blanchard, G., Noiret, C.: System identifiability (symbolic computation) and parameter estimation (numerical computation). Numerical Algorithms 34, 282–292 (2003)
14. Boulier, F., Denis-Vidal, L., Henin, T., Lemaire, F.: LÉPISME. In: Proceedings of the ICPSS conference (2004); Submitted to the Journal of Symbolic Computation, http://hal.archives-ouvertes.fr/hal-00140368
15. Boulier, F.: Differential Elimination and Biological Modelling. Radon Series on Computational and Applied Mathematics (Gröbner Bases in Symbolic Analysis), vol. 2, pp. 111–139 (October 2007), http://hal.archives-ouvertes.fr/hal-00139364
16. Ritt, J.F.: Differential Algebra. Dover Publications Inc, New York (1950), http://www.ams.org/online_bks/coll33
17. Kolchin, E.R.: Differential Algebra and Algebraic Groups. Academic Press, New York (1973)
18. Wang, D.: Elimination Practice: Software Tools and Applications. Imperial College Press, London (2003)
19. Hairer, E., Wanner, G.: Solving ordinary differential equations II. Stiff and Differential–Algebraic Problems, 2nd edn. Springer Series in Computational Mathematics, vol. 14. Springer, New York (1996)
20. Boulier, F., Lazard, D., Ollivier, F., Petitot, M.: Representation for the radical of a finitely generated differential ideal. In: ISSAC 1995: Proceedings of the 1995 international symposium on Symbolic and algebraic computation, pp. 158–166. ACM Press, New York (1995), http://hal.archives-ouvertes.fr/hal-00138020
21. Boulier, F., Lemaire, F.: A computer scientist point of view on Hilbert's differential theorem of zeros. Algebra in Engineering, Communication and Computing (submitted, 2007), http://hal.archives-ouvertes.fr/hal-00170091
22. Kokotovic, P., Khalil, H.K., O'Reilly, J.: Singular Perturbation Methods in Control: Analysis and Design. Classics in Applied Mathematics 25 (1999)

23. Van Breusegem, V., Bastin, G.: Reduced order dynamical modelling of reaction systems: a singular perturbation approach. In: Proceedings of the 30th IEEE Conference on Decision and Control, Brighton, England, pp. 1049–1054 (December 1991)
24. Vora, N., Daoutidis, P.: Nonlinear model reduction of chemical reaction systems. AIChE Journal 47(10), 2320–2332 (2001)
25. Bennet, M.R., Volfson, D., Tsimring, L., Hasty, J.: Transient Dynamics of Genetic Regulatory Networks. Biophysical Journal 92, 3501–3512 (2007)
26. Maquet, F.: Master Research training report. University Lille I (to appear, 2008)
27. Boulier, F., Lefranc, M., Lemaire, F., Morant, P.E., Ürgüplü, A.: On proving the absence of oscillations in models of genetic circuits. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) Ab 2007. LNCS, vol. 4545, pp. 66–80. Springer, Heidelberg (2007), http://hal.archives-ouvertes.fr/hal-00139667
28. Boulier, F., Lefranc, M., Lemaire, F., Morant, P.E.: Applying a rigorous quasi-steady state approximation method for proving the absence of oscillations in models of genetic circuits. In: Algebraic Biology 2008 (submitted, 2008), http://hal.archives-ouvertes.fr/hal-00213327
29. Niu, W., Wang, D.: Algebraic Approaches to Stability Analysis of Biological Systems. Mathematics in Computer Science 1, 507–539 (2008)
30. Noiret, C.: Utilisation du calcul formel pour l'identifiabilité de modèles paramétriques et nouveaux algorithmes en estimation de paramètres. PhD thesis, Université de Technologie de Compiègne (2000)
31. Boulier, F.: The BLAD libraries (2004), http://www.lifl.fr/~boulier/BLAD
32. Walter, É.: Identifiability of State Space Models. Lecture Notes in Biomathematics, vol. 46. Springer, Heidelberg (1982)
33. Ollivier, F.: Le problème de l'identifiabilité structurelle globale : approche théorique, méthodes effectives et bornes de complexité. PhD thesis, École Polytechnique, Palaiseau, France (1990)
34. Diop, S., Fliess, M.: Nonlinear observability, identifiability, and persistent trajectories. In: Proc. 30th CDC, Brighton, pp. 714–719 (1991)
35. Ljung, L., Glad, S.T.: On global identifiability for arbitrary model parametrisations. Automatica 30, 265–276 (1994)
36. Sedoglavic, A.: A Probabilistic Algorithm to Test Local Algebraic Observability in Polynomial Time. Journal of Symb. Comp. 33(5), 735–755 (2002)
37. Boulier, F., Lemaire, F., Moreno Maza, M.: Computing differential characteristic sets by change of ordering. Technical report, Université Lille I (2007); Submitted to the Journal of Symbolic Computation, http://hal.archives-ouvertes.fr/hal-00141095
38. Mishra, B.: Algebra, Automata, Algorithms & Beyond. Le Matematiche LXIII (1), 21–23 (2008)

Hybrid Semantics for Stochastic π-Calculus

Luca Bortolussi[1] and Alberto Policriti[2]

[1] Dept. of Mathematics and Computer Science, University of Trieste, Italy
luca@dmi.units.it
[2] Dept. of Mathematics and Computer Science, University of Udine,
and Applied Genomics Institute, Udine Italy
policriti@dimi.uniud.it

Abstract. We put forward a method to map stochastic π-calculus processes in chemical ground form into hybrid automata, a class of dynamical systems with both discrete and continuous evolution. The key ingredient is the separation of control and molecular terms, which turns out to be related to the conservation properties of the system.

1 Introduction

Systems biology is a fertile research area in which experimental techniques are coupled with mathematical modeling in order to understand the complex dynamics within cells [16]. In this context, both mathematical and computational tools play an important role: biological systems must be described in a precise mathematical framework, usually ordinary differential equations or stochastic processes, and the obtained models must then be analyzed. However, due to their intrinsic complexity, computational techniques like simulation must be used.

The contributions of Computer Science, on the other hand, are not just restricted to the computational analysis of models, but also related to their description by means of suitable formal languages, offering the features of *compositionality* and *model reusability* [19].

These languages usually belong to the domain of stochastic process algebras (SPA), which have a semantics defined in terms of Continuous Time Markov Chains [14,21]. Different SPA have been used in biological modeling; here we recall stochastic π-calculus [18], PEPA [6], and stochastic Concurrent Constraint Programming (sCCP, [5]). An important issue in using these languages is that the resulting models are automatically interpreted as stochastic processes, hence they can be simulated and analyzed using common techniques, like the celebrated Gillespie's algorithm [11]. Actually, when just biochemical reactions are involved, the stochastic process defined by these SPA coincide with the canonical one, as given by the chemical master equation [11,21]. In this approach, the system is described *microscopically*, counting the number of molecules of each species.

Simulation of stochastic processes, however, can be computationally too demanding, especially when large populations of chemical species are present in the system. In this case, a better strategy is that of using models based on ordinary differential equations, approximating the number of molecules as a continuous

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 40–55, 2008.

quantity. In order to achieve this goal, PEPA, stochastic π-calculus, and sCCP have been equipped with a semantics based on ODE's [3,15,8].

This sort of approximation, however, is not appropriate for models having small populations and for models containing inherently discrete entities like genes (genes are usually present in a single copy in a cell, and they are neither produced nor degraded). Actually, there are several well-known examples in which the stochastic model and its continuous approximation show a radically different behavior [11,2]. In order to deal with some of these cases, in [4] we introduced a hybrid approximation for sCCP, in which some entities of the system are kept discrete, while others are approximated as continuous. Essentially, we defined a mapping from sCCP to hybrid automata, which are mixed discrete/continuous dynamical systems. The separation between continuous and discrete components in sCCP is quite natural. Each sCCP program consists of a set of agents acting on shared variables. Each agent can have different internal states, enabling different actions, while variables describe time-changing quantities. For instance, proteins are described by variables, while genes are modeled as agents, with different internal states depending on which transcription factors are bound to them (see [5] for further details). Therefore, the discrete dynamics of the hybrid automata associated with an sCCP program comes from the different inner states of agents, while the continuous dynamics describes the time evolution of sCCP variables. In [4] it is shown that the hybrid semantics of sCCP is more adherent to the stochastic dynamics than the continuous one.

The purpose of this paper is to extend the method of [4] to stochastic π-calculus. In this case, we do not have any *a-priori* distinction between discrete agents and time-varying variables as in sCCP, hence we need to separate π-agents into two groups, one amenable to continuous approximation and the other inherently discrete. This distinction turns out to be related to conservation properties of the system.

The paper is organized as follows: Section 2 recalls the syntax of a subset of stochastic π-calculus that will be used in the following, while Section 3 briefly presents Hybrid Automata. The identification of discrete components of π-calculus programs is tackled in Section 4, while Section 5 presents the mapping to Hybrid Automata. Conclusions are drawn in Section 6.

2 Stochastic π-Calculus

We recall briefly the syntax of stochastic π-calculus processes in Chemical Ground Form (CGF from now on), as defined in [7]. This is a restricted subset of π-calculus (actually, of CCS), which is however sufficient to model biochemical reactions and genetic networks. Essentially, the restriction operator is dropped and no information is passed on channels during a communication. Parameterized communication can however be introduced without substantially changing the language, see [7] for further details.

Processes in CGF are defined by the grammar of Table 1. Essentially, a π-calculus program in CGF consists of a set of reagents (agents/molecules) E and

Table 1. Syntax of stochastic π-calculus processes in Chemical Ground Form

$$
\begin{array}{c}
E ::= \mathbf{0} | X = M, E \\
M ::= \mathbf{0} | \pi.P \oplus M \\
P ::= \mathbf{0} | (X | P) \\
\pi ::= \tau^r | ?x^r | !x^r \\
CGF ::= (E, P)
\end{array}
$$

of the initial solution (network/configuration) P_0.[1] Each reagent is a summation M, whose addends are guarded by actions, either silent (τ^r) or communications on channels (inputs $?x^r$ and outputs $!x^r$). All actions are indexed by a parameter r, which is the rate of an exponential distribution governing their execution time. It is agreed that all occurrences of each channel have the same rate, so that the function ρ, assigning to each distinct channel name and silent action its rate, can be consistently defined.

Remark 1. The same channel name x can be used several times in the definition of reagents in E, so that different agent types can communicate on it. For instance, consider $X_1 = !x.X_1$, $X_2 = ?x.X_2$, $X_3 = !x.X_3$, and $X_4 = ?x.X_4$. x gives rise to 4 possible communications (X_1 with X_2 or X_4 and $X_3$with X_2 or X_4). In general, if there are n occurrences of $!x^r$ and m occurrences of $?x^r$, then there are mn possible interactions on channel x. Actually, an equivalent system can be easily constructed where all possible communications have distinct names: we can replace x with nm new channels $x^r_{i,j}$, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$, substituting the i^{th} occurrence of $!x^r.P$ with the sum $!x^r_{i,1}.P \oplus \ldots \oplus !x^r_{i,m}.P$, and the j^{th} occurrence of $?x^r$ with $!x^r_{1,j}.P \oplus \ldots \oplus !x^r_{n,j}.P$. Note that this can cause a quadratic explosion in the description of $CGF(E, P)$, although it leaves the dynamics unchanged.

From now on, we confine ourselves to programs in which each channel appears once as input and once as output. In addition, we index all τ actions with a different integer number, in such a way that τ actions are all distinct.

Example 1. We present here a very simple genetic regulatory network, consisting of one single gene producing a protein that represses its own production. For simplicity, we consider a model in which the gene generates directly the proteins, without the intermediate step of mRNA production. This system can be described by the following set of reactions:

$$
\begin{array}{c}
G + P \rightarrow_{k_b} G_b + P \\
G_b \rightarrow_{k_u} G \\
G \rightarrow_{k_p} G + P \\
P \rightarrow_{k_d} \emptyset,
\end{array}
\tag{1}
$$

where G is the gene, G_b is the repressed gene and P is the produced protein.

[1] Each solution P can be compactly described by its "marking", i.e. by counting the number of copies of each agent X in parallel in P. This is sufficient to determine the dynamics of the system.

Following the conversion rules from chemical reactions prescribed in [7], the π-calculus program in CGF associated to (1) is defined as

$$\begin{aligned} G &= ?b^{k_b}.G_b \oplus \tau_p^{k_p}.(G|P) \\ G_b &= \tau_u^{k_u}.G \\ P &= !b^{k_b}.P \oplus \tau_d^{k_d}.\mathbf{0} \end{aligned} \tag{2}$$

In this encoding, each molecule is represented as a distinct process, and its action capabilities correspond to the different reactions of which the molecule is a reactant. A full program requires also the specification of the initial state of the system, which in this case consists of one single copy of G.

Associating ODE's with π-calculus programs in CGF. Given a π-calculus program (E, P_0) in CGF, we can associate with it a set of ODE's quite straightforwardly. Our presentation differs slightly from [7,8], in order to simplify the following discussion.

First, we need some preliminary definitions. The function $action(X)$ returns the set of actions of the reactant X; it can be extended to set of agents by $action(\{X_1, X_2\}) = action(X_1) \cup action(X_2)$. For instance, $action(P) = \{b, \tau_d\}$, where P is defined in (2), and $action(\{P, G_b\}) = \{b, \tau_d, \tau_u\}$. Therefore, $action(E)$ is the set of actions of all agents defined in E. In addition, with $\#(X, P)$ ($\#(X, B)$) we denote the number of occurrences of the agent X in the solution P (multiset B). Finally, with $react(\pi)$ and $prod(\pi)$ we indicate the multisets of agents that are consumed and produced by π, respectively. For instance, if $X = !x.(X|X)$ and $Y = ?x.(Y|Y)$, then $react(x) = [\![X, Y]\!]$ and $prod(x) = [\![X, X, Y, Y]\!]$.

The basic idea in associating ODE's with a π-program is to approximate the number of occurrences $\#(X, P)$ of an agent X in the solution P by a continuous quantity, also indicated with X ($\mathbf{X}$, instead, will denote the vector of all variables X).

Definition 1. *Let (E, P_0) be a π-calculus program in CGF, $A \subseteq E$, and $T \subseteq action(E)$.*

1. *The* stoichiometric matrix *$S_{A,T}$ w.r.t. A and T is a $|A| \times |T|$ matrix, with rows indexed by agents of A and columns indexed by actions of T, defined by $S_{A,T}[X, \pi] = \#(X, prod(\pi)) - \#(X, react(\pi))$.*
2. *The rate vector $\phi_{A,T}$ w.r.t. A and T is a $|T|$ vector defined by*[2]

$$\phi_{A,T}[\pi](\mathbf{X}) \begin{cases} 0, & \text{if } react(\pi) \cap A = \emptyset; \\ \rho(\pi)X, & \text{if } react(\pi) \cap A = [\![X]\!]; \\ \rho(\pi)XY, & \text{if } react(\pi) \cap A = [\![X, Y]\!]; \\ \rho(\pi)X(X-1), & \text{if } react(\pi) \cap A = [\![X, X]\!]. \end{cases}$$

[2] The rate function for homodimeric reactions lacks a factor 1/2 w.r.t. the standard definition. This happens because each homodimer $X = !x.X \oplus ?x.X$ counts twice: once for the input $!x$ and once for the output $?x$. This gives a factor two in the rate function canceling out 1/2. Clearly, this must be taken into account while writing models in π-calculus: rates of homodimeric reactions must be halved [7].

3. *The ODE's associated to* (E, P_0) *are* $\dot{\mathbf{X}} = S_{E,action(E)} \cdot \phi_{E,action(E)}(\mathbf{X})$, *with initial conditions given by* $X(0) = \#(X, P_0)$.[3]

Consider again the program of Example 1. For $T = action(E)$, we obtain:

$$S_{E,T} = \begin{matrix}(G)\\(G_b)\\(P)\end{matrix}\begin{pmatrix}-1 & 0 & 0 & 1\\ 1 & 0 & 0 & -1\\ 0 & 1 & -1 & 0\end{pmatrix} \quad \phi_{E,T} = \begin{pmatrix}\rho(b)GP\\ \rho(\tau_p)G\\ \rho(\tau_d)P\\ \rho(\tau_u)G_b\end{pmatrix} \quad \begin{cases}\dot{G} = \rho(\tau_u)G_b - \rho(b)GP\\ \dot{G_b} = \rho(b)GP - \rho(\tau_u)G_b\\ \dot{P} = \rho(\tau_p)G - \rho(\tau_d)P\end{cases}$$

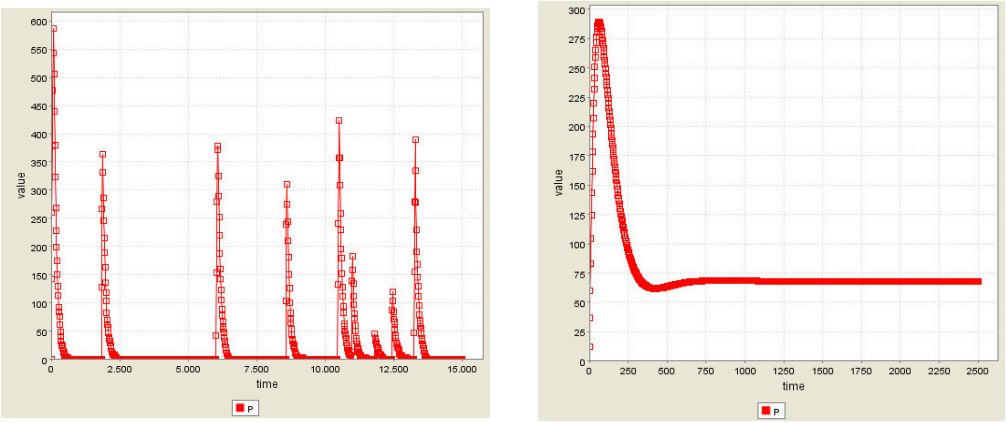
In Figure 1 we compare a stochastic simulation of the π-calculus model (2) with the numerical solution of the associated ODE's. As we can readily see, the two plots are different. In particular, in the stochastic simulation, P is produced in bursts and it follows an irregular oscillatory pattern. The ODE's system, instead, presents a much simpler pattern of evolution, in which the quantity of P converges to an asymptotic value. This divergence is caused by the fact that, approximating continuously the state of the gene, we lose any information on the discrete dynamics of gene's activations and deactivations. As a matter of fact, the same loss occurs when we consider the average trajectory of the stochastic system. In fact, the average presents a trend more similar to that of Figure 1(b) (data not shown), not conveying an adequate description of the dynamics. Consider, for instance, an event triggered when the concentration of P exceeds 100: this event would be activated infinitely often in the stochastic system, but just once in the ODE-based one (or in the average trajectory).

Remark 2. In the mapping from π-calculus to ODE's of Definition 1, the rates of the stochastic processes and of the ODE's are the same, in contrast with the usual praxis in biochemistry, in which rates of ODE's are redefined in terms of concentrations. Indeed, the ODE's that we defined must be thought of as an approximation of the stochastic process, in the sense that they are a first-order approximation of the differential equation for the average of the stochastic system, cf. [1]. This relationship is similar to the one connecting deterministic and stochastic mass action models of chemical reactions, cf. [10].

3 Hybrid Automata

In this section we briefly recall the ideas and the definition of hybrid automaton. The reader is referred to [13] for an introductory survey. Hybrid automata are dynamical systems presenting both discrete and continuous evolution. Essentially, they are defined using a set of variables evolving continuously in time, subject to instantaneous changes induced by the happening of discrete *control* events. When discrete events happen the automaton enters its next *mode*, where the laws governing the flow of continuous variables change. Formally, a hybrid automaton is a tuple $H = (V, E, \mathbf{X}, flow, init, inv, jump, reset)$, where:

[3] We could have defined the stoichiometric matrix and the rate vector just for the sets E and $action(E)$, instead of parameterizing them w.r.t. subsets A and T. This parametric version, however, will turn out to be useful in Section 5.



(a) π-calculus model of system (1) (b) ODE model of system (1)

Fig. 1. (a) Simulation of the π-calculus model (2). The red line corresponds to P. Parameters of the models are the following: $k_p = 1$, $k_b = 0.0001$, $k_u = 0.0005$, $k_d = 0.01$. The initial configuration consists in a single copy of G. (b) numerical simulation of ODE's associated to the π-calculus model (2), for the same parameters just given. The evolution of P is different from the stochastic case, as it converges quickly to an asymptotic value. Parameters have been assigned in order to correspond to a situation in which the binding/unbinding dynamics of the repressor P is very slow, when compared to protein production and degradation. Increasing the binding and unbinding strength smooths the behavior of the stochastic system, making it closer to Figure 1(b). We refer the reader to [4] for a more biologically relevant example.

- $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a finite set of real-valued variables (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).
- $G = (V, E)$ is a finite labeled graph, called *control graph*. Vertices $v \in V$ are the *(control) modes*, while edges $e \in E$ are called *(control) switches* and model the happening of a discrete event.
- Associated with each vertex $v \in V$ there is a set of ordinary differential equations[4] $\dot{\mathbf{X}} = flow(v)$ (referred to as the *flow conditions*). Moreover, $init(v)$ and $inv(v)$ are two formulae on $\mathbf{X}$ specifying the *admissible initial conditions* and some *invariant conditions* that must be true during the continuous evolution of variables in v (forcing a change of mode to happen when violated).
- Edges $e \in E$ of the control graph are labeled by $jump(e)$, a formula on $\mathbf{X}$ stating for what values of variables each transition is active (the so called *activation region*), and by $reset(e)$, a formula on $\mathbf{X} \cup \mathbf{X}'$ specifying the change of the variables' values after the transition has taken place.

The *traces* of the system are defined as the time traces of the continuous variables. Notice that the activation conditions are in general non-deterministic (as well as resets), hence there can be different traces starting from the same initial values. In this paper we are concerned mainly with *simulation* of hybrid automata, i.e. with the generation of a set of admissible traces.

[4] Other forms of flow specification are possible (differential inclusions, first order formulae, etc.) but sets of differential equations are sufficient for our purposes here.

In the following, we will need a product construction for HA, which is almost the classical one [13], the only difference being the treatment of fluxes for variables shared among the factors. In our case, in fact, fluxes are added. Before giving the formal definition of this *flux product*, we put forward some notation. The product $G = G_1 \times G_2$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ has vertex set $V_1 \times V_2$ and edges of the form $((v_1, w), (v_2, w))$, where $(v_1, v_2) \in E_1$, or $((v, w_1), (v, w_2))$, where $(w_1, w_2) \in E_2$. Given an edge $e \in E$, the projection $\pi_1(e)$ is defined for all edges $e = ((v_1, w), (v_2, w))$, and is the edge $(v_1, v_2) \in E_1$. Projection π_2 can be defined symmetrically.

Definition 2. *Let* $H_1 = (V_1, E_1, \mathbf{X_1}, flow_1, init_1, inv_1, jump_1, reset_1)$ *and* $H_2 = (V_2, E_2, \mathbf{X_2}, flow_2, init_2, inv_2, jump_2, reset_2)$. *The* flux product *of* H_1 *and* H_2 *is the hybrid automaton* $H_1 \otimes H_2 = (V, E, \mathbf{X}, flow, init, inv, jump, reset)$ *defined by:*

1. $(V, E) = (V_1, E_1) \times (V_2, E_2)$;
2. $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$;
3. $flow((v_1, v_2)) \restriction_X = flow_1(v_1) \restriction_X + flow_2(v_2) \restriction_X$, *if* $X \in \mathbf{X_1} \cap \mathbf{X_2}$. *Otherwise, if* $X \in \mathbf{X_i}$, *then* $flow((v_1, v_2)) \restriction_X = flow_i(v_i) \restriction_X$;
4. $init((v_1, v_2)) = init_1(v_1) \wedge init_2(v_2)$ *and* $inv((v_1, v_2)) = inv_1(v_1) \wedge inv_2(v_2)$;
5. $jump(e) = jump_1(e_1)$, *if* $e \in E$ *is such that* $\pi_1(e) = e_1$, *otherwise, if* $\pi_2(e) = e_2$, *then* $jump(e) = jump_2(e_2)$;
6. $reset(e) = reset_1(e_1)$ *if* $\pi_1(e) = e_1$, *while* $reset(e) = reset_2(e_2)$ *if* $\pi_2(e) = e_2$.

4 Control Automata

The reactants E of a π-calculus program (E, P_0) in CGF can be broadly separated into two classes, one comprising all those terms which will change in quantity during the evolution (essentially, molecules) and the other containing a collection of terms defining a *control structure*, in such a way that exactly one term of the collection is active in every solution P reachable from P_0. For instance, consider the model of the gene of Example 1. The gene G has two mutually exclusive states: the normal form G and the repressed form G_b. Obviously, we always have a single occurrence of the gene in the system, which can be in one of the two states G and G_b. Essentially, we can think of a gene as a kind of "logical" entity, whose activity depends on its inner state and whose state may change due to interactions with the system.

We want now to find suitable conditions to define collections of terms behaving like the gene in the example, which will be called in the rest of the paper *control automata* (CA), because they can be thought of as automata present in a single copy and synchronizing with the rest of the system. Let R be a CA. First, as the reagents $X_i \in R$ represent different inner states of the CA, there should not be any inner communication between two of them:

$$\forall X \in R, \forall \pi \in action(X), react(\pi) = [\![X, Y]\!] \Rightarrow Y \notin R. \tag{3}$$

In addition, any action involving $X \in R$ (call it an R-state) can change the state, but in any case it cannot destroy nor create more than one R-state in the current solution. Hence R must satisfy:

$$\forall X \in R, \forall \pi \in action(X), \exists! Y \in R : Y \in prod(\pi). \tag{4}$$

A further condition that must be satisfied, is that only actions involving one component of R may change its state:

$$\forall \pi \in action(E), react(\pi) \cap R = \emptyset \Rightarrow prod(\pi) \cap R = \emptyset. \tag{5}$$

The final request on R is that in the initial configuration P_0 of CGF, exactly one state of R is present in P_0 (we indicate $\#(X, P_0)$ with $\#_0(X)$):

$$\#_0(R) = \sum_{X \in R} \#_0(X) = 1. \tag{6}$$

Collecting all the previous conditions together, we are ready to define a control automaton:

Definition 3. *Let (E, P_0) be a π-calculus program in CGF. A subset $R \subseteq E$ is a* control automaton *if and only if it satisfies properties (3), (4), (5), (6).*

A minimal control automaton *R is a CA such that no proper subset of R is a CA. $\mathcal{C}(E)$ denotes the set of all minimal CA of E.*

The conditions satisfied by CA imply that the number of occurrences of elements of R in the system remains constant during each possible computation, in fact equal to one. Formally, we define

Definition 4. *A set $R \subseteq E$ of reagents is* conserved *iff $\#(R, P) = \#(R, P')$, for every pair of solutions P, P' such that $P \xrightarrow{\pi} P'$ (i.e. P can be transformed into P' by the action π), where $\#(R, P) = \sum_{X \in R} \#(X, P)$.*

Theorem 1. *$R \subseteq E$ is a CA $\Rightarrow$ R is conserved.*

Proof. Let P, P' be two configurations such that $P \xrightarrow{\pi} P'$. Clearly, the following relation holds:

$$\#(R, P') = \#(R, P) - \#(R, react(\pi)) + \#(R, prod(\pi)). \tag{7}$$

If $react(\pi) \cap R = \emptyset$, condition (5) implies that $prod(\pi) \cap R = \emptyset$, hence $\#(R, P') = \#(R, P)$ by equation (7). Otherwise, condition (3) implies that $\#(R, react(\pi)) = 1$, while condition (4) states that $\#(R, prod(\pi)) = 1$, hence $\#(R, P') = \#(R, P)$ again by equation (7). ■

Corollary 1. *If $R \subseteq E$ is a CA, then, for each configuration P reachable from P_0, $\#(R, P) = 1$.* ■

The previous theorems state that control automata are indeed entities with different states, one of which only is active at each stage of the evolution of the system: control automata are neither produced nor degraded. This results in a conservation law, in which the sum of the quantity of internal states of the automata always equals one. Conservation laws allow us to use linear algebra to characterize set of conserved states, as we will show in the following.

Before that, we observe that the previous theorem cannot be reversed. Consider this simple system:

$$X = \tau_1.Y \oplus !x.X \quad Y = \tau_2.X \oplus ?x.Y,$$

where X can change state into Y and viceversa, due to a silent action. Moreover, X and Y can synchronize on channel x, a condition violating property (3). Note that all actions τ_1, τ_2, x maintain constant the quantity $\#(X,P)+\#(Y,P)$. If no other action of the system can create or destroy copies of X and Y, clearly $R = \{X, Y\}$ is conserved. R, however, is not a CA, as it fails to satisfy properties (3) and (4). If the starting configuration of the system is such that $\#_0(R) = 1$, then no communication on x will ever be possible, as we will never have a pair of X, Y agents ready to communicate. In this case, we may remove the branch $!x.X$ from X and the branch $?x.Y$ on Y without altering the behavior of the system. The resulting agents X' and Y' are now CA. This justifies the following definition:

Definition 5. *Let $R \subseteq E$. The reduced form $\overline{R}$ of R is obtained by removing all occurrences of channels x such that $react(x) \cap R = \{X, Y\}$.*

We can now prove the following:

Theorem 2. *If $R \subseteq E$ is conserved and $\#_0(R) = 1$, then the reduced form $\overline{R}$ of R is a control automaton.*

Proof. If R is conserved, then for every action $\pi \in \mathcal{A}$

$$\#(R, react(\pi)) = \#(R, prod(\pi)) \tag{8}$$

From this equation, property (5) follows immediately. In addition, property (4) holds for all π such that $\#(R, react(\pi)) = 1$. If no $\pi \in action(E)$ is such that $\#(R, react(\pi)) = 2$, then the set R is a CA. Otherwise, it becomes a CA after removing all $\pi \in action(E)$ is such that $\#(R, react(\pi)) = 2$, i.e. all inner communications in R. The resulting set is exactly the reduced form $\overline{R}$ of R. ■

Theorems 1 and 2 give us a criterion to separate terms belonging to Control Automata (i.e. agents exerting a control activity in the system) from agents whose number changes over time. In fact, we need to identify collections R of reagents that are conserved in the evolution of the system, whose initial quantity is $\#_0(R) = 1$.

Remark 3. The characterization of CA as conserved sets has been suggested to us by the study of conservation properties of Petri Nets [21]. As a matter of fact, it is possible to define a mapping of stochastic π-calculus in CGF to Petri Nets and reason on the latter. However, working directly with π-calculus is more intuitive in this context.

Theorem 3. *Let (E, P_0) be a π-calculus program in CGF. A set R is conserved iff the vector $\mathbf{y}_R$ on E, equal to 1 for $X \in R$ and to 0 otherwise, belongs to the null space of the matrix $A = S^T_{E,action(E)}$.*

Proof. Let $\mathbf{r}$ be a non-negative vector indexed by $action(E)$. If $\mathbf{r}$ represents the set of actions happening in a solution P with multiplicity of each term given by a vector $\mathbf{x}$ on E (i.e. $x_i = \#(X_i, P)$), then the configuration after the happening of $\mathbf{r}$ has multiplicity $\mathbf{x}' = \mathbf{x} + S_{E,action(E)}\mathbf{r}$ (remember that the stoichiometric matrix $S_{E,action(E)}$ gives the net variation of each reagent in E after the happening of each action of $action(E)$). Suppose now $A\mathbf{y}_R = 0$, and $\mathbf{x}$, $\mathbf{x}'$ are such that $\mathbf{x}' = \mathbf{x} + S_{E,action(E)}\mathbf{r}$ for some $\mathbf{r}$. The number of reagents of R in $\mathbf{x}$ is given by $\sum_{\mathbf{y}_R[X]=1} \mathbf{x}[X] = \mathbf{y}_R^T\mathbf{x}$. The net variation of R after $\mathbf{r}$ is

$$\mathbf{y}_R^T\mathbf{x} - \mathbf{y}_R^T\mathbf{x}' = \mathbf{y}_R^T(\mathbf{x} - \mathbf{x}') = \mathbf{y}_R^T(S_{E,action(E)}\mathbf{r}) = (A\mathbf{y}_R)^T\mathbf{r} = 0,$$

hence R is conserved. Viceversa, if R is conserved, then $(A\mathbf{y}_R)^T\mathbf{r} = 0$ for all $\mathbf{r}$, hence $A\mathbf{y}_R = 0$. ■

The previous property gives a clear way to identify Control Automata. All we have to do is find all conserved sets R, i.e. vector of the null space of A whose entries are either zero or one, whose initial value in P_0 is one. Essentially, we have to solve a *zero-one integer programming problem* under the constraints $A\mathbf{y} = 0$ and $\sum_{X\in E} \#_0(X)\mathbf{y}[X] = 1$.

Suppose now we have solved this problem and collected the set of solutions. In this way, we have identified all the candidates to be control automata. Some of them satisfy Definition 4 only after removing inner communications that cannot fire, hence they are CA contingent on the initial conditions. Accepting them as proper CA would make the construction of the next section dependent on initial conditions. To avoid this, we simply remove these pseudo-CA from the solution set. Formally, let $\mathbf{y}_R$ be the 0-1 E-vector that is equal to 1 for agents in R and 0 elsewhere; we consider the set $\mathcal{R}(E) = \{R \subseteq E \mid A \cdot \mathbf{y}_R = 0 \wedge \sum_{X\in R} \#_0(X) = 1\}$ and remove the pseudo-CA obtaining $\mathcal{R}^-(E) = \{R \in \mathcal{R}(E) \mid \forall \pi \in action(E), \#(R, react(\pi)) \leq 1\}$. This new set has some nice closure properties:

Proposition 1. *Let $R, R_1, R_2 \in \mathcal{R}^-(E)$ such that $R_1 \subset R$ and $R_2 \subset R$. Then $R_1 \cap R_2 \neq \emptyset$ and $R_1 \cap R_2 \in \mathcal{R}^-(E)$.* ■

This allows to define consistently the minimal representative $\mu_E(R)$ of a CA $R \in \mathcal{R}^-(E)$ as the intersection of all subsets contained in R:

$$\mu_E(R) = \bigcap_{R'\in\mathcal{R}^-(E), R'\subseteq R} R'.$$

Proposition 1 implies that $\mu_E(R) \in \mathcal{R}^-(E)$, hence the set of *minimal CA* $\mathcal{R}_M(E) = \{\mu_E(R) \mid R \in \mathcal{R}^-(E)\}$ is a subset of $\mathcal{R}^-(E)$. Actually, $\mathcal{R}_M(E)$ is precisely the set we are interested in, according to the following theorem.

Theorem 4. *Let (E, P_0) be in CGF. Then, $\mathcal{C}(E) = \mathcal{R}_M(E)$.* ■

Let's go back to our running example. In this case, the stoichiometric matrix is equal to

$$S_{E,action(E)} = \begin{matrix}(G)\\(G_b)\\(P)\end{matrix}\begin{pmatrix}-1 & 0 & 0 & 1\\ 1 & 0 & 0 & -1\\ -1 & 1 & -1 & 0\end{pmatrix}$$

and the initial state of the system consists only of one copy of G. It is immediate to verify that the only solution to $S^T_{E,actionE}\mathbf{y} = \mathbf{0}$ and $\mathbf{y}[G] = 1$ is $\mathbf{y} = (1, 1, 0)$, corresponding to the CA $R = \{G, G_b\}$. This is exactly the set of inner states of the gene, in agreement with the intuitions at the beginning of this section.

Consider now the following π-program: $X = \tau.(X_1|X_2)$, $X_1 = !x.Y_1$, $X_2 = ?x.Y_2$, $Y_1 = \tau_1.X_1$, $Y_2 = \tau_2.X_2$. It holds that $\mathcal{R}_M = \{R_1, R_2\}$, with $R_1 = \{X, X_1, Y_1\}$ and $R_2 = \{X, X_2, Y_2\}$. However, these sets are not disjoint, in contrast with the intuition of control automata as distinct objects. Actually, R_1 and R_2 can be "separated" simply replacing X with $X^1_0|X^2_0$, $X^1_0 = !z.X_1$ and $X^2_0 = ?z.X_2$, where $\rho(z) = \rho(\tau)$, an operation not altering the dynamics. Although now $R_1 = \{X^1_0, X_1, Y_1\}$ and $R_2 = \{X^2_0, X_2, Y_2\}$ are disjoint, their evolution is still entangled, as they need to synchronize on z to evolve. In order to keep the mapping to HA simple, we suppose in the following that the set $\mathcal{C}(E)$ of minimal CA of E satisfy the following two conditions:

Definition 6. *Let (E, P_0) be a π-calculus program in CGF with control automata $\mathcal{C}(E)$.*

1. *$\mathcal{C}(E)$ is* independent *iff for all $R_1, R_2 \in \mathcal{C}(E)$, $R_1 \cap R_2 = \emptyset$.*
2. *$\mathcal{C}(E)$ is* non-interacting *iff no two CA of $\mathcal{C}(E)$ can synchronize on an action $\pi \in action(E)$, i.e. there does not exist $R_1, R_2 \in \mathcal{C}(E)$, $\pi \in action(E)$ such that $react(\pi) \cap R_1 \neq \emptyset \wedge react(\pi) \cap R_2 \neq \emptyset$.*

Remark 4. The problem of finding if a 0-1 integer programming problem has a feasible solution is known to be $\mathcal{NP}$-complete [9]. This makes the use of the method defined above potentially troublesome in terms of computational cost. In addition, there is no a-priori bound on the size of the set of solutions $\mathcal{R}(E)$ that needs to be processed to obtain $\mathcal{R}_M(E)$. Therefore, we need to study the complexity of the problem of determining the set of minimal CA, giving, if possible, more refined algorithms. We leave further investigation of these issues for the future.

5 From π-Calculus to Hybrid Automata

In order to map a π-calculus program (E, P_0) into CGF to a Hybrid Automaton (HA), we process independently each minimal Control Automaton R of (E, P_0), associating a HA to R. Then, these HA are combined together using the flux product construction of Definition 2. In the following, we suppose that (E, P_0) has independent and non-interacting CA (cf. Definition 6).

Before giving the details of the mapping, we need to fix some preliminary notation. Let $R \in \mathcal{C}(E)$ be a CA of E. The set of *discrete actions of R* is $T_d(R) = \{\pi \in action(E) \mid react(\pi) \cap R \neq \emptyset \wedge react(\pi) \cap prod(\pi) \cap R = \emptyset\}$, i.e. the set of actions changing the state of R. Similarly, the set of *continuous actions of*

R is $T_c(R) = \{\pi \in action(E) \mid react(\pi) \cap prod(\pi) \cap R \neq \emptyset\}$, i.e. the set of actions looping on a state of R. Given an action $\pi \in T_d(R) \cup T_c(R)$, its starting state is $start(\pi, R) = X$ iff $react(\pi) \cap R = \{X\}$. Similarly, if $prod(\pi) \cap R = \{Y\}$, then the ending state of π is $end(\pi, R) = Y$. With $T_d(R, X) = \{\pi \in T_d(R) \mid start(\pi, R) = X\}$ we indicate the subset of discrete transitions of R starting at X; a similar definition applies to $T_c(R, X)$. The set T_C of *uncontrolled continuous actions* is $T_C = action(E) \setminus \bigcup_{R \in \mathcal{R}_M(E)} (T_d(R) \cup T_c(R))$. Finally, the *continuous terms* of (E, P_0) are those not belonging to a CA, i.e. $Cont(E) = \{X \in E \mid X \notin \bigcup_{R \in \mathcal{C}(E)} R\}$.

Consider now a CA $R \in \mathcal{C}(E)$. The HA associated to R will be $H(R) = (V(R), E(R), \mathbf{X}, flow, init, inv, jump, reset)$. We discuss how to define its components separately.

Control Graph. The discrete modes $V(R)$ of the HA are simply the different elements of the CA R, i.e. $V(R) = \{\sigma_X \mid X \in R\}$. The edges of the control graph are those implied by the discrete transitions $T_d(R)$: for $\pi \in T_d(R)$, we add the edge $(start(\pi, R), end(\pi, R))$.

Continuous Flow. The (continuous) variables $\mathbf{X}$ of the system are one for each term in $Cont(E)$. These variables, in a state $\sigma_X \in V(R)$, are subject only to the effect of continuous transitions $T_c(R, X)$. The idea to define their continuous flow is simply to restrict the method of Definition 1 to transitions $T_c(R, X)$ for each X in R. Essentially, we define the local stoichiometric matrix $S_{Cont(E),T_c(R,X)}$ and the local rate vector $\phi_{Cont(E),T_c(R,X)}$ according to Definition 1, and we set $flow(\sigma_X) = S_{Cont(E),T_c(R,X)} \cdot \phi_{Cont(E),T_c(R,X)}(\mathbf{X})$. Note that, according to Definition 1, $\phi_{Cont(E),T_c(R,X)}$ depends only on the variables in $Cont(E)$.

The invariant conditions are not needed in our mapping, hence $inv(\sigma_X) = true$, while initial conditions are defined according to the initial configuration P_0 of (E, P_0): if $\#_0(X) = 0$, then $init(\sigma_X) := false$, while if $\#_0(X) = 1$ then $init(\sigma_X) := \bigwedge_{Y \in Cont(E)} (Y = \#_0(Y))$.

Discrete Transitions. The most delicate part of the mapping is the definition of the activation conditions and of the resets of discrete transitions. The point is that in the stochastic process associated to (E, P_0), transitions of $T_d(R)$ are stochastic, with execution time exponentially distributed according to their rate. This stochastic duration needs to be removed taking into account the *timing* of the corresponding event. The simplest possibility is that of *firing the transition at its expected time*. There is however a complication given by the fact that the rate of the transition can depend on some terms that are approximated as continuous. The associated stochastic process is, therefore, a *non-homogeneous Poisson process* [20] with time-dependent rate $\phi_{Cont(E),action(E)}[\pi](\mathbf{X}(t)) = \phi_\pi(\mathbf{X}(t))$. In this case, given the *cumulative rate* at time t, $\Lambda(t) = \int_0^t \phi_\pi(\mathbf{X}(s))ds$, the probability of firing within time t is $F(t) = 1 - e^{-\Lambda(t)}$. The expected value of $\Lambda(T)$ at the time T of firing is $E[\Lambda(T)] = 1$ (cf. [4,20]). Hence, we can fire the transition whenever $\Lambda(t) \geq 1$. In order to describe this condition within HA framework, we introduce a new (clock) variable Z_π, with initial value 0, evolving according to

the equation $\dot{Z}_\pi = \phi_\pi(\mathbf{X})$. When the transition fires, we reset all variables Z_π to zero, due to memoryless property of CTMCs.

The jump predicate can thus be defined as follows. If $\pi \in T_d(R, X)$ is such that $react(\pi) = \{X\}$, then $jump(\pi) := Z_\pi \geq 1$. Else, if $react(\pi) = \{X, Y\}$, then $jump(\pi) := Z_\pi \geq 1 \wedge Y \geq 1$. The fact that CA are non-interacting guarantees that $Y \in Cont(E)$. Moreover, each discrete transition will be *urgent*, meaning that it will fire as soon as its guard becomes true.

Resets of discrete transitions need to take into account the modifications induced on continuous variables. Moreover, they need to set to zero variables Z_π. Hence, for $\pi \in T_d(R, X)$, we set $reset(\pi) := \bigwedge_{Y \in Cont(E)}(Y' = Y + \#(Y, prod(\pi)) - \#(Y, react(\pi))) \wedge \bigwedge_{\pi_1 \in T_d(R)}(Z'_{\pi_1} = 0)$.

We collect now the previous discussion into a formal definition.

Definition 7. *Let $R \in \mathcal{C}(E)$ be a CA of a π-calculus program (E, P_0) in CGF. $H(R) = (V(R), E(R), \mathbf{W}, flow, init, inv, jump, reset)$, the Hybrid Automaton associated to R, is defined by:*

1. *$V(R) = \{\sigma_X \mid X \in R\}$ and $E(R) = \{(start(\pi, R), end(\pi, R)) \mid \pi \in T_d(R)\}$;*
2. *$\mathbf{W} = \mathbf{X} \cup \mathbf{Z}$, where $\mathbf{X}$ is the vector of variables on $Cont(E)$ and $\mathbf{Z} = \{Z_\pi \mid \pi \in T_d(R)\}$;*
3. *for each $\sigma_X \in V(R)$, $flow(\sigma_X)[Y] := S_{Cont(E),T_c(R,X)}[Y, \cdot]\phi_{Cont(E),T_c(R,X)}$ for $Y \in \mathbf{X}$ and $flow(\sigma_X)[Z_\pi] := \phi_{Cont(E),action(E)}[\pi]$ for $Z_\pi \in \mathbf{Z}$;*
4. *for each $\sigma_X \in V(R)$, $inv(\sigma_X) := true$;*
5. *for each $\sigma_X \in V(R)$, $init(\sigma_X) := false$ if $\#_0(X) = 0$ and $init(\sigma_X) := \bigwedge_{Y \in Cont(E)}(Y = \#_0(Y)) \wedge \bigwedge_{\pi \in T_d(R)}(Z_\pi = 0)$ if $\#_0(X) = 1$;*
6. *for each $\pi \in T_d(R)$, $jump(\pi) := (Z_\pi \geq 1) \wedge \bigwedge_{Y \in react(\pi) \cap Cont(E)}(Y \geq 1)$;*
7. *for each $\pi \in T_d(R)$, $reset(\pi) := \bigwedge_{Y \in Cont(E)}(Y' = Y + \#(Y, prod(\pi)) - \#(Y, react(\pi))) \wedge \bigwedge_{\pi_1 \in T_d(R)}(Z'_{\pi_1} = 0)$.*

The previous definition gives us the recipe to associate an hybrid automaton to each control automaton of a π-calculus program. However, there are some actions that are not taken into account up to now, namely those belonging to the set of uncontrolled continuous actions T_C. In order to take into account their effect, we define a special hybrid automaton with one single state.

Definition 8. *Let (E, P_0) be a π-calculus program in CGF and T_C the set of uncontrolled continuous transitions. The Hybrid Automaton $H(T_C)$ associated to T_C is $H(T_C) = (V, E, \mathbf{X}, flow, init, inv, jump, reset)$, where $V = \{\sigma\}$, $E = \emptyset$, $jump := false$, $reset := false$, $inv(\sigma) := true$, $init(\sigma) := \bigwedge_{X \in Cont(E)} X = \#_0(X)$, and $flow(\sigma) := S_{Cont(E),T_C}\phi_{Cont(E),T_C}(\mathbf{X})$, where $\mathbf{X}$ is the vector of variables on $Cont(E)$.*

We are finally ready to define the HA associated to the whole π-program: we simply need to take the flux product of the HA coming from the different CA of $\mathcal{C}(E)$ and of the HA coming from uncontrolled actions.

Definition 9. *Let (E, P_0) be a π-calculus program in CGF with a non-interacting and independent $\mathcal{C}(E)$ of CA. The Hybrid Automaton $H(E)$ associated to (E, P_0) is*

$$H(E) = H(T_C) \otimes \bigotimes_{R \in \mathcal{C}(E)} H(R).$$

Consider again the simple autoregulatory gene network of Example 1. According to the computation at the end of Section 4, the system has one control automata, namely $R = \{G, G_b\}$, and one continuous term P. The Hybrid Automata $H(R)$ and $H(T_C)$ are shown in Figure 2, together with their product $H(E)$. We can see that the final HA has three continuous variables, one corresponding to P and the other two, Z_b, Z_{τ_u}, associated to the discrete edges $T_d(R) = \{b, \tau_u\}$. We can also see how the final vector field for P is the sum of the vector fields for P of the two factors. In Figure 3, we show a simulation of $H(E)$ for the same parameters as in Figure 1. As we can see, the dynamics of the hybrid automaton resembles very closely the behavior of the stochastic system rather than the one of the ODE's, preserving alternating spikes. Hence, keeping part of the discreteness of the original system was enough to maintain qualitatively the oscillatory pattern of Figure 1(a). We stress the fact that the correspondence

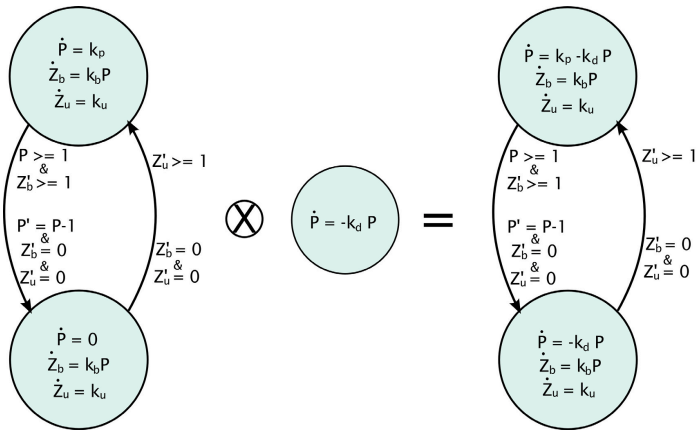


Fig. 2. Hybrid Automata obtained for the π-calculus process of Example 1

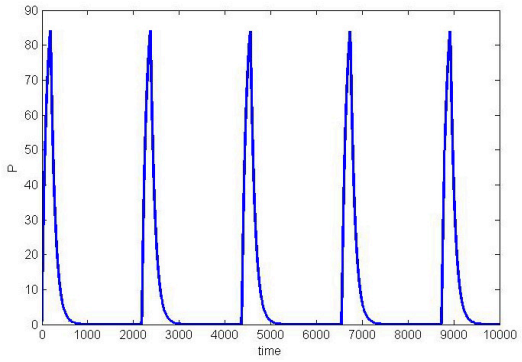


Fig. 3. Simulation of the Hybrid Automaton of Figure 2, for the same parameters as in Figure 1

between the plots of Figures 1(a) and 3 is only partially quantitative. For instance, in both the stochastic and the hybrid model P overcomes the threshold of 100 infinitely often. As a matter of fact, the noise of the stochastic systems perturbs dramatically the period of oscillations, which is highly regular in the hybrid system, due to the urgent policy for transitions. In [4] we suggested the use of non-determinism to reintroduce a certain degree of variability.

Remark 5. In the definition of the mapping, we assumed that $\mathcal{C}(E)$ is independent and non-interacting. The choice of independence, i.e. of having disjoint minimal CA, is fundamental in order to use the flux product construction. Non-interactivity, instead, can be dropped by introducing a product of HA that allows synchronization of discrete transitions. However, we did not follow this direction here, to keep the presentation simpler.

6 Conclusions

In this paper we provided a restricted form of stochastic π-calculus, the so called Chemical Ground Form [7], with a semantics based on hybrid automata. The most difficult aspect is the identification, without the availability of external knowledge, of portions of a π-calculus program acting as discrete controlling components. This is necessary to define the discrete skeleton of the HA and it is related to conservation laws of the system.

We argued that the hybrid semantics is more appropriate to reproduce qualitatively the stochastic behavior of the simple genetic network of Example 1. The same observation seems to apply to a broad class of genetic circuits, cf. [4] for further details.

Comparing qualitatively the dynamical evolution of systems described with different mathematical formalisms is itself an intriguing problem. We believe that a reasonable choice can be based on a temporal logic to describe dynamical properties, thus equating behavioral equivalence with equi-satisfiability.

As a matter of fact, Hybrid Automata may be used also to tackle the discreteness given by molecules present in small quantities in the system. The idea is roughly that of separating actions into two groups, those amenable of continuous approximation and those to be kept discrete. Different partitions correspond to different HA, *de facto* associating to each π-calculus model a lattice of HA, differing in the degree of discreteness. Dynamic partition schemes can be considered as well. This approach will result in a flexible hybrid semantics, capable of dealing effectively with size effects and multi-scale systems. Another important issue consists in stating a clear connection between our hybrid semantics and standard models of biochemical networks, like the chemical master equation (CME) or mass action ODEs. A possibility in this sense is to manipulate the CME similarly to [12], where authors formally justify hybrid simulation schemes mixing stochastic differential equations with discrete jump Markov processes. However, the advantage of our hybrid semantics with respect to hybrid

simulation strategies like [17,12] is that HA offer a powerful and flexible framework both for simulation and for verification, computationally more efficient than stochastic processes.

References

1. Bortolussi, L.: A master equation approach to differential approximations of stochastic concurrent constraint programming. In: Proceedings of QAPL 2008 (2008)
2. Bortolussi, L., Policriti, A.: Dynamical systems and stochastic programming I - ordinary differential equations. Trans. of Comp. Sys. Bio. (submitted, 2008)
3. Bortolussi, L., Policriti, A.: Stochastic concurrent constraint programming and differential equations. In: Proceedings of QAPL 2007. ENTCS, vol. 16713 (2007)
4. Bortolussi, L., Policriti, A.: Hybrid approximation of stochastic concurrent constraint programming. In: Proceedings of IFAC 2008 (2008)
5. Bortolussi, L., Policriti, A.: Modeling biological systems in concurrent constraint programming. Constraints 13(1) (2008)
6. Calder, M., Gilmore, S., Hillston, J.: Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. Trans. of Comp. Sys. Bio. 4230, 1–23 (2006)
7. Cardelli, L.: From processes to ODEs by chemistry (2006), `http://lucacardelli.name/`
8. Cardelli, L.: On process rate semantics. In: TCS (2007)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: a Guide to the Therory of NP-Completeness. Freeman, New York (1979)
10. Gillespie, D.: The Chemical Langevin Equation. Jo. of Chem. Phys. 113(1), 297–306 (2000)
11. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. of Phys. Chem. 81(25) (1977)
12. Haseltine, E.L., Rawlings, J.B.: On the origins of approximations for stochastic chemical kinetics. J. Chem. Phys. 123 (2005)
13. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings of LICS 1996 (1996)
14. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
15. Hillston, J.: Fluid flow approximation of PEPA models. In: Proceedings of QEST 2005 (2005)
16. Kitano, H.: Computational systems biology. Nature 420, 206–210 (2002)
17. Neogi, N.A.: Dynamic partitioning of large discrete event biological systems for hybrid simulation and analysis. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 463–476. Springer, Heidelberg (2004)
18. Priami, C., Quaglia, P.: Modelling the dynamics of biosystems. Briefings in Bioinformatics 5(3), 259–269 (2004)
19. Regev, A., Shapiro, E.: Cellular abstractions: Cells as computation. Nature 419 (2002)
20. Ross, S.M.: Stochastic Processes. Wiley, New York (1996)
21. Wilkinson, D.J.: Stochastic Modelling for Systems Biology. Chapman & Hall, Boca Raton (2006)

Applying a Rigorous Quasi-Steady State Approximation Method for Proving the Absence of Oscillations in Models of Genetic Circuits

François Boulier[1], Marc Lefranc[2], François Lemaire[1], and Pierre-Emmanuel Morant[2]

[1] University Lille I, LIFL, 59655 Villeneuve d'Ascq, France
{Francois.Boulier,Francois.Lemaire}@lifl.fr
http://www.lifl.fr/~{boulier,lemaire}
[2] University Lille I, PHLAM, 59655 Villeneuve d'Ascq, France
Marc.Lefranc@univ-lille1.fr, morant@phlam.univ-lille1.fr
http://www-phlam.univ-lille1.fr/perso/lefranc

Abstract. In this paper, we apply a rigorous quasi-steady state approximation method on a family of models describing a gene regulated by a polymer of its own protein. We study the absence of oscillations for this family of models and prove that Poincaré-Andronov-Hopf bifurcations arise if and only if the number of polymerizations is greater than 8. A result presented in a former paper at *Algebraic Biology 2007* is thereby generalized. The rigorous method is illustrated over the basic enzymatic reaction.

1 Introduction

In a former paper [1], we studied a simple family of models depending on an integer parameter n and featuring a negative feedback loop, one of the core ingredients for generating oscillations [2]. These abstract models are closely related to models studied by Goodwin and Griffith in the 60's [3,4,5]. Griffith considered a model of a gene regulated by a polymer formed of n copies of its own protein. We studied the same problem, but in a slightly more general case, where gene activation is not assumed to be fast. We eventually concluded with the absence of Poincaré-Andronov-Hopf bifurcation in our family of models for $n \leq 8$ and their existence for $n \geq 9$. The absence/presence of Poincaré-Andronov-Hopf bifurcation for $n \leq 8$ is a strong indicator for the absence/presence of oscillating trajectories. Extensive numerical experiments [6,7] confirmed the absence of oscillations for $n \leq 8$ and their existence for $n \geq 9$.

In this paper, the models are designed by means of systems of parametric nonlinear ordinary differential equations (ODE) [8]. The approach applied in [1] consisted in two steps: first simplifying the initial system of $n + 2$ parametric ordinary differential equations as a reduced system of three ODE by means of a quasi-steady state approximation; second, studying the reduced model.

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 56–64, 2008.

The idea of quasi-steady state approximation is simple: study the dynamics of the slow reactions, assuming that the fast ones are at quasi-equilibrium, thereby removing from the ODE system, the differential equations which describe the evolution of the variables at quasi-equilibrium. Many authors [9,10,11,12] state that carrying out rigorously this approximation is far from straightforward. We would rather say that there are different ways to perform this approximation and that it is the problem of ascertaining the domain of validity of each kind of approximation which is not straightforward.

In another paper [13], the authors reformulated the methods of [9,10,11,12], which are equivalent, and made them fully algorithmic, by means of differential elimination methods [14,15,16,17,18]. An efficient implementation, based on [19], was developed by the third author.

In this paper, we show that the reduction method of [13] can be applied to our family of models. It yields a reduced model which contains that of [1] as a particular case: our new approximation is more precise. By a very concise proof, we show that the results obtained in [1] also hold for the new model. This paper gives us also the opportunity to widen the audience of our [13, DifferentialModelReduction] algorithm by recalling its principle.

2 Our Family of Models

2.1 The Initial Model

A schematic model describing a single gene regulated by an order n polymer of its own protein is provided in Fig. 1. It is borrowed from [1, page 68]. The variables G and H represent the state of the gene. The mRNA concentration and the concentration of the protein translated from the mRNA are represented by M and P. The n types of polymers of P are denoted by $P = P_1, P_2, \ldots, P_n$. Greek letters represent parameters.

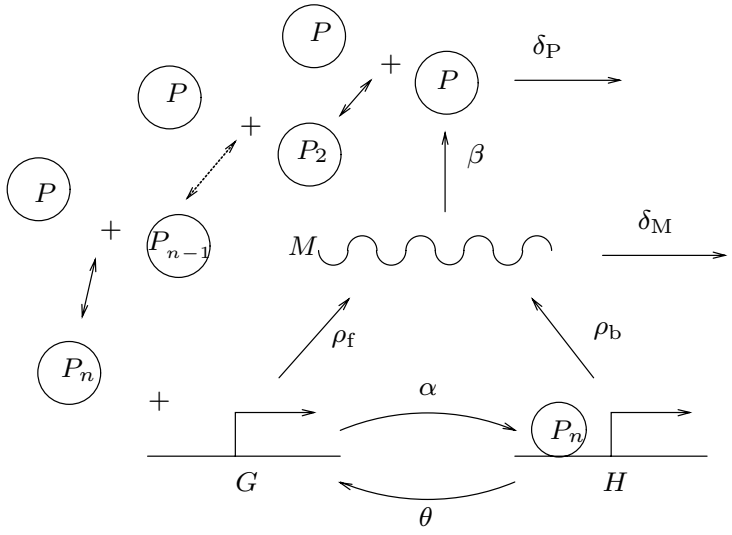


Fig. 1. A single gene regulated by a polymer of its protein

First, one translates the diagram as a system of generalized [20] chemical reactions (transcription and translation are not balanced reactions), introducing $2\,(n-1)$ extra parameters k_i^-, k_i^+ $(1 \leq i \leq n-1)$:

$$G + P_n \underset{\theta}{\overset{\alpha}{\rightleftharpoons}} H, \quad G \xrightarrow{\rho_\mathrm{f}} G + M, \quad H \xrightarrow{\rho_\mathrm{b}} H + M,$$
$$M \xrightarrow{\beta} M + P, \quad M \xrightarrow{\delta_\mathrm{M}} \emptyset, \quad P \xrightarrow{\delta_\mathrm{P}} \emptyset, \qquad P_i + P \underset{k_i^-}{\overset{k_i^+}{\rightleftharpoons}} P_{i+1} \quad (1 \leq i \leq n-1).$$

The initial model is obtained by translating the above chemical reactions system as the following system of parametric ordinary differential equations, denoting $A_i = (k_i^-\, P_{i+1} - k_i^+\, P_i\, P)$. Variables G, H, M, $P = P_1, \ldots, P_n$ are time varying functions. The dot appearing over the variables, in the equations left handsides, denotes the derivative w.r.t. the time.

$$\begin{aligned}
\dot{G} &= \theta\, H - \alpha\, G\, P_n, \\
\dot{H} &= -\theta\, H + \alpha\, G\, P_n, \\
\dot{M} &= \rho_\mathrm{f}\, G + \rho_\mathrm{b}\, H - \delta_\mathrm{M}\, M, \\
\dot{P} &= \beta\, M - \delta_\mathrm{P}\, P + 2\, A_1 + A_2 + \cdots + A_{n-1}, \\
\dot{P}_i &= -A_{i-1} + A_i \qquad (2 \leq i \leq n-1), \\
\dot{P}_n &= -A_{n-1} + \theta\, H - \alpha\, G\, P_n.
\end{aligned} \tag{1}$$

The variables G and H should be viewed as "random variables" instead of concentrations. They are bound by the relation $G + H = \gamma_0$ where γ_0 is a constant equal to the total quantity of the gene. In [1], the variable H was replaced by $\gamma_0 - G$.

2.2 The New Quasi-Steady State Approximation

The quasi-steady state approximation that is performed in this paper relies on the following hypotheses: the $n-1$ chemical reactions describing the polymerization of the protein are fast while the other ones are slow. This happens when the parameters k_i^+, k_i^- are much larger than the other parameters.

Given those hypotheses, one applies the algorithm [13, DifferentialModelReduction] on our model (Fig. 1). The principle of the method is recalled in Sect. 4.1. Applied on our model, the algorithm amounts to the following steps. First one replaces, in the initial model, the contributions of the fast reactions by new variables F_i $(1 \leq i \leq n-1)$. This just amounts to rewriting $A_i = F_i$ in system (1). Then one adds the following algebraic equations to the system, in order to express the pre-equilibrium conditions:

$$0 = k_i^+\, P\, P_i - k_i^-\, P_{i+1} \qquad (1 \leq i \leq n-1).$$

Then one eliminates the new variables F_i from the above differential-algebraic system.[1] One is led to the raw reduced model:

[1] Our algorithm does not handle the generic system with a symbolic n. We computed the reduced system for many different values of n, inferred the general formula and, checked afterwards that the inferred formula is correct.

$$\begin{aligned}
\dot{G} &= \theta\, H - \alpha\, K_{n-1}\, P^n\, G,\\
\dot{H} &= -\theta\, H + \alpha\, K_{n-1}\, P^n\, G,\\
\dot{M} &= \rho_{\mathrm{b}}\, H + \rho_{\mathrm{f}}\, G - \delta_{\mathrm{M}}\, M,\\
\dot{P} &= \frac{n\,\theta\, H - n\,\alpha\, K_{n-1}\, P^n\, G - \delta_{\mathrm{P}}\, P + \beta\, M}{\displaystyle\sum_{i=0}^{n-1} (i+1)^2\, K_i\, P^i}
\end{aligned} \tag{2}$$

where $K_i = \dfrac{k_1^+ \cdots k_i^+}{k_1^- \cdots k_i^-}$ with the convention $K_0 = 1$.

2.3 Parameters Reduction

The raw reduced model (2) can now be simplified by rescaling all parameters and variables. The following equations express the old variables as functions of the new ones, which are overlined:

$$\theta = \overline{\theta}\,\overline{\delta_{\mathrm{M}}}, \quad \beta = \overline{\beta}\,\overline{\delta_{\mathrm{M}}}, \quad \delta_{\mathrm{P}} = \overline{\delta}\,\overline{\delta_{\mathrm{M}}}, \quad \rho_{\mathrm{b}} = \frac{\overline{\mu}\,\overline{\delta}\,\overline{\theta}\,\overline{\delta_{\mathrm{M}}}}{\overline{\alpha}\,\overline{\beta}}, \quad \rho_{\mathrm{f}} = \frac{\overline{\delta}\,\overline{\theta}\,(\overline{\mu}+\overline{\lambda})\,\overline{\delta_{\mathrm{M}}}}{\overline{\alpha}\,\overline{\beta}},$$

$$\delta_{\mathrm{M}} = \overline{\delta_{\mathrm{M}}}, \quad \alpha = \frac{\overline{\alpha}\,\overline{\delta_{\mathrm{M}}}}{\overline{K}_{n-1}}, \quad K_i = \frac{\overline{K}_i\,\overline{\alpha}^i}{\overline{\theta}^i} \quad (1 \le i \le n-1),$$

$$G = \overline{G}, \quad H = \overline{\gamma_0} - \overline{G}, \quad M = \frac{\overline{M}\,\overline{\delta}\,\overline{\theta}}{\overline{\alpha}\,\overline{\beta}}, \quad P = \frac{\overline{\theta}\,\overline{P}}{\overline{\alpha}}, \quad t = \frac{\overline{t}}{\overline{\delta_{\mathrm{M}}}}.$$

Performing these substitutions in the raw reduced model (2), discarding the redundant ODE which expresses the evolution of H and removing the bars for legibility, one gets the reduced model (3):

$$\begin{aligned}
\dot{G} &= \theta\,(\gamma_0 - G - G\,P^n),\\
\dot{M} &= \lambda\, G + \gamma_0\,\mu - M,\\
\dot{P} &= \frac{n\,\alpha\,(\gamma_0 - G - G\,P^n) + \delta\,(M - P)}{\displaystyle\sum_{i=0}^{n-1} (i+1)^2\, K_i\, P^i}.
\end{aligned} \tag{3}$$

Remark 1. In order to recover [1, system (1)], it is sufficient to replace the right handside of the last equation of system (3) by its numerator.

Last observe that one more parameter could be removed from the above system, the software [21] shows. The above reduction is however more convenient in this paper for it permits us to directly apply the results of [1].

3 On the Existence of Poincaré-Andronov-Hopf Bifurcations

We prove in this section that no Poincaré-Andronov-Hopf bifurcation [22] arises in system (3) for meaningful[2] values of the parameters and the variables if and only if $n \leq 8$. Our proof essentially amounts to reducing the study of system (3) to that of [1, system (1)] and then applying the main result of [1].

According to remark 1, the steady point equations of system (3) are exactly those of [1, Sect. 4]. They write:

$$\gamma_0 = G + G\,P^n, \quad M = P, \quad \lambda = \frac{P - \mu\,G - \mu\,G\,P^n}{G}.$$

The Jacobian matrix of system (3), evaluated at the steady points[3] writes:

$$J = \begin{pmatrix} -\theta\,(1+P^n) & -n\,\theta\,G\,P^{n-1} & 0 \\ \dfrac{P - \mu\,G - \mu\,G\,P^n}{G} & 0 & -1 \\ -\dfrac{n\,\alpha\,(1+P^n)}{B} & -\dfrac{n^2\,\alpha\,G\,P^{n-1} + \delta}{B} & \dfrac{\delta}{B} \end{pmatrix}$$

where $B = \sum_{i=0}^{n-1} (i+1)^2\,K_i\,P^i$.

If one clears the denominators of the last row of the Jacobian matrix J, i.e. if one lets $B = 1$, then one exactly gets the Jacobian matrix of [1, page 73].

Remark 2. The matrix J is invariant under the following transformation, where ℓ denotes a nonzero arbitrary constant:

$$B \to \ell\,B, \quad \delta \to \ell\,\delta, \quad \alpha \to \ell\,\alpha.$$

Remark 3. The parameters K_i only occur in the denominator B of the Jacobian matrix J.

Proposition 1. *A Poincaré-Andronov-Hopf bifurcation arises for [1, system (1)] if and only if such a bifurcation arises for the system (3), for meaningful values of the systems variables and parameters.*

Proof. Assume that a Poincaré-Andronov-Hopf bifurcation occurs for system (3), for some *meaningful*[4] values of the parameters and variables ($1 \leq i \leq n-1$):

$$(G,\, P,\, M,\, \lambda,\, \alpha,\, \theta,\, \delta,\, \gamma_0,\, \mu,\, K_i) = (G^0,\, P^0,\, M^0,\, \lambda^0,\, \alpha^0,\, \theta^0,\, \delta^0,\, \gamma_0^0,\, \mu^0,\, K_i^0).$$

[2] Following [1], all variables and parameters are required to be positive apart λ, which may be negative but must anyway be greater than $-\mu$.

[3] Note that the derivative of B w.r.t. to P which appears in the Jacobian of the system (3) disappears after the evaluation at the steady points since it is multiplied by a term that cancels it.

[4] In the sense precised above.

Then, by introducing $B^0 = \sum_{i=0}^{n-1}(i+1)^2\,K_i^0\,(P^0)^i$, and according to the two remarks above, a bifurcation of [1, system (1)] occurs for the meaningful values:

$$(G,\,P,\,M,\,\lambda,\,\alpha,\,\theta,\,\delta,\,\gamma_0,\,\mu,) = (G^0,\,P^0,\,M^0,\,\lambda^0,\,\alpha^0/B^0,\,\theta^0,\,\delta^0/B^0,\,\gamma_0^0,\,\mu^0).$$

Conversely, suppose that a birfurcation of [1, system (1)] arises for some meaningful values

$$(G,\,P,\,M,\,\lambda,\,\alpha,\,\theta,\,\delta,\,\gamma_0,\,\mu) = (G^0,\,P^0,\,M^0,\,\lambda^0,\,\alpha^0,\,\theta^0,\,\delta^0,\,\gamma_0^0,\,\mu^0).$$

Using the two remarks above and taking $\ell = 2$, one can easily find some positive values for $K_1^0, \ldots, K_{n-1}^0$ such that $B_0 = \sum_{i=0}^{n-1}(i+1)^2\,K_i^0\,(P^0)^i = 2$. Thus a bifurcation for the system (3) occurs when

$$(G,\,P,\,M,\,\lambda,\,\alpha,\,\theta,\,\delta,\,\gamma_0,\,\mu,\,K_i) = (G^0,\,P^0,\,M^0,\,\lambda^0,\,2\alpha^0,\,\theta,\,2\delta^0,\,\gamma_0^0,\,\mu^0,\,K_i^0)$$

which are meaningful values. □

The next proposition follows from the results of [1] and Prop. 1:

Proposition 2. *For meaningful values of the parameters and the variables, no Poincaré-Andronov-Hopf bifurcation arises in system (3) if and only if* $n \leq 8$.

4 On the New Quasi-Steady State Approximation

4.1 Principle of the Method

Our method is based on similar ideas as in [11,9,12] (see [13, Sect. 4.2] for more details). The main interest of our method is that it is fully algorithmic. One first builds a system of differential equations involving some one extra variable F_i for each fast reaction. Those extra variables are then eliminated by performing elimination. Our original implementation is based on the DIFFALG [23] package available in the standard library of MAPLE. A more recent implementation is based on the RegularChains [19] package.

This section aims at summarizing [13, Sect. 2]. Consider the classical system of chemical reactions (4) and (5) describing the transformation of a substrate S into a product P under the action of the enzyme E (an intermediate complex C is produced):

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \tag{4}$$

$$C \xrightarrow{k_2} E + P \tag{5}$$

Assume that the reaction (4) is fast. Our method consists in introducing the following system:

$$\Big[\dot{E} = -F_1 + k_2\,C,\quad \dot{S} = -F_1,\quad \dot{C} = F_1 - k_2\,C,\quad \dot{P} = k_2\,C,\quad k_1\,E\,S = k_{-1}\,C\Big].$$

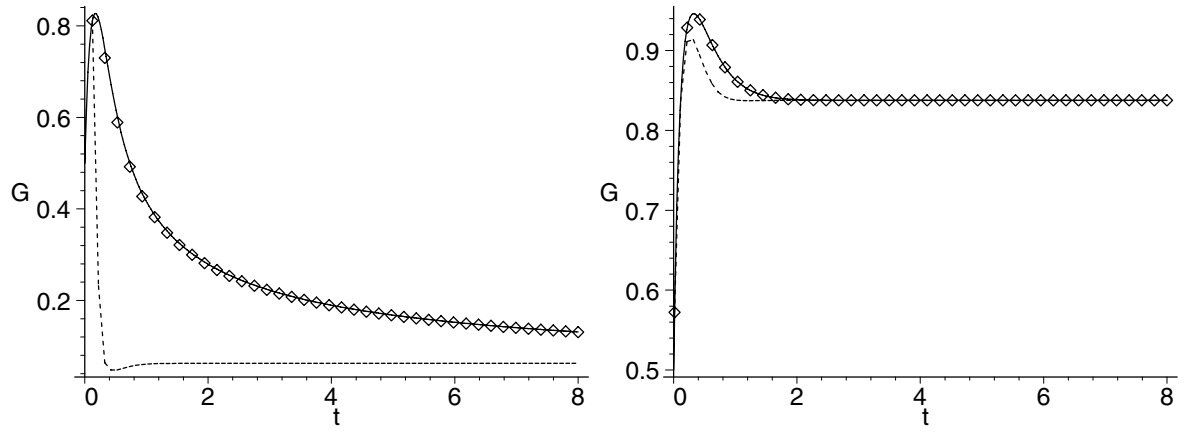


Fig. 2. Numerical simulations of the variable $G(t)$. The horizontal axis give the time t. The vertical one gives the value of G. For both simulations, $n = 3$, $G(0) = 0.5$, $M(0) = 1$, $P(0) = 1$, $\gamma_0 = 1$, $\alpha = 1$, $\theta = 10$, $\rho_{\mathrm{f}} = 10$, $\rho_{\mathrm{b}} = 5$, $\beta = 50$, $\delta_{\mathrm{M}} = 5$, $\delta_{\mathrm{P}} = 10$ and $k_i^+ = 100$ for each i. On the leftmost picture, $k_i^- = 100$ thus $K_i = (100/100)^i = 1$ for each i. On the rightmost one, $k_i^- = 2000$ thus $K_i = (100/2000)^i = 0.05^i$ for each i. The solid line (original model (1)) and the diamond dots (reduced model (2)) curves almost coincide. The dotted line [1, last system page 69] curve is the lower one on both pictures.

In this system, the new variable F_1 denotes the rate of reaction (4). This variable introduces one degree of freedom. However, this freedom is contrained by the the quasi-steady state equilibrium of reaction (4) which is $k_1\,E\,S = k_{-1}\,C$.

The value of F_1 can be computed by performing an elimination process which yield:

$$\left[F_1 = \frac{k_2\,E\,S\,(S+K)}{K\,(S+E+K)},\ \dot{E} = \frac{k_2\,E^2\,S}{K\,(S+E+K)},\ \dot{P} = \frac{k_2\,E\,S}{K},\ \dot{S} = -\frac{k_2\,E\,S\,(S+K)}{K\,(S+E+K)},\ C = \frac{E\,S}{K}\right]$$

where $K = k_{-1}/k_1$.

Using the conservation laws $E + C = E_0 + C_0$ and $S + C + P = S_0 + C_0 + P_0$ (where the subscript 0 indicates the initial concentration), assuming that $C_0 = P_0 = 0$ and introducing $V_m = k_2\,E_0$, further computations yield:

$$\dot{S} = -\frac{V_m\,S\,(K+S)}{K\,E_0 + (K+S)^2} \tag{6}$$

which differs from the Henri-Michaëlis-Menten and Briggs-Haldane formulae[5]:

$$\dot{S}(t) = -\frac{V_m\,S(t)}{K + S(t)}. \tag{7}$$

With this easy example, the benefits are clear since the reduction is automatic and yields the formula (6) which seems more accurate than the classical reductions, especially when the condition $S \gg E_0$ is not fulfilled. Observe that formula (7) is recovered from (6) by assuming $S \gg E_0$.

[5] $K = \frac{k_{-1}}{k_1}$ in Henri-Michaëlis-Menten's case, $K = \frac{k_{-1}+k_2}{k_1}$ in Briggs-Haldane's.

4.2 A Better New Reduced Model

The reduced system (2) appears to be more precise than that of [1], as numerical simulations show. Figure 2 shows two different numerical simulations of the variable $G(t)$ for the same parameters and initial conditions values except for the k_i^-'s (hence for the K_i's). In both cases, three curves are displayed: one for the initial model (1) in solid line, one for the raw reduced model (2) in diamond dots and one for [1, last system page 69] in dotted line. In both cases, diamond dots are almost superimposed on the solid line.

The reduction performed in [1] is clearly less precise than the new one. As the K_i's parameters tend towards zero, the reduction performed in [1] becomes more accurate (the rightmost picture is obtained with values of K_i's smaller than that of the leftmost one). Thus the domain of validity of the reduction performed in [1, last system page 69] is narrower than that of our new reduction: the K_i's need to be small.

5 Conclusion

In this paper, the result presented in [1] is generalized by applying an algorithmic, accurate, quasi-steady state approximation method.

It is well-known that quasi-steady state approximation is useful for it permits to reduce the size of the differential system to study and the number of its parameters. But quasi-steady state approximation could also be viewed as a way to study the dynamical properties of gene regulatory networks which are invariant for a range of reactions mechanisms since the mechanisms involved in the fast reactions may not need to be known in detail. This issue is important [24]. This shows that the development of algorithmic and accurate quasi-steady state approximation methods is an important research domain in the field of algebraic biology.

References

1. Boulier, F., Lefranc, M., Lemaire, F., Morant, P.E., Ürgüplü, A.: On proving the absence of oscillations in models of genetic circuits. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) Ab 2007. LNCS, vol. 4545, pp. 66–80. Springer, Heidelberg (2007), http://hal.archives-ouvertes.fr/hal-00139667
2. Fall, C.P., Marland, E.S., Wagner, J.M., Tyson, J.J.: Computational Cell Biology. Interdisciplinary Applied Mathematics, vol. 20. Springer, Heidelberg (2002)
3. Goodwin, B.C.: Temporal Organization in Cells. Academic Press, London (1963)
4. Goodwin, B.C.: Advances in Enzyme Regulation, vol. 3, p. 425. Pergamon Press, Oxford (1965)
5. Griffith, J.S.: Mathematics of Cellular Control Processes. I. Negative Feedback to One Gene. Journal of Theoretical Biology 20, 202–208 (1968)
6. Doedel, E.: AUTO software for continuation and bifurcation problems in ODEs (1996), http://indy.cs.concordia.ca/auto

7. Ermentrout, B.: Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students. Software, Environments, and Tools, vol. 14. SIAM, Philadelphia (2002)
8. Conrad, E.D., Tyson, J.J.: Modeling Molecular Interaction Networks with Nonlinear Differential Equations. In: Szallasi, Z., Stelling, J., Periwal, V. (eds.) System Modeling in Cell Biology: From Concepts to Nuts and Bolts, pp. 97–124. The MIT Press, Cambridge (2006)
9. Van Breusegem, V., Bastin, G.: Reduced order dynamical modelling of reaction systems: a singular perturbation approach. In: Proceedings of the 30th IEEE Conference on Decision and Control, Brighton, England, pp. 1049–1054 (December 1991)
10. Okino, M.S., Mavrovouniotis, M.L.: Simplification of Mathematical Models of Chemical Reaction Systems. Chemical Reviews 98(2), 391–408 (1998)
11. Vora, N., Daoutidis, P.: Nonlinear model reduction of chemical reaction systems. AIChE Journal 47(10), 2320–2332 (2001)
12. Bennet, M.R., Volfson, D., Tsimring, L., Hasty, J.: Transient Dynamics of Genetic Regulatory Networks. Biophysical Journal 92, 3501–3512 (2007)
13. Boulier, F., Lefranc, M., Lemaire, F., Morant, P.E.: Model Reduction of Chemical Reaction Systems using Elimination. In: The international conference MACIS 2007 (2007), http://hal.archives-ouvertes.fr/hal-00184558
14. Ritt, J.F.: Differential Algebra. Dover Publications Inc, New York (1950), http://www.ams.org/online_bks/coll33
15. Kolchin, E.R.: Differential Algebra and Algebraic Groups. Academic Press, New York (1973)
16. Boulier, F., Lazard, D., Ollivier, F., Petitot, M.: Representation for the radical of a finitely generated differential ideal. In: ISSAC 1995: Proceedings of the 1995 international symposium on Symbolic and algebraic computation, pp. 158–166. ACM Press, New York (1995), http://hal.archives-ouvertes.fr/hal-00138020
17. Wang, D.: Elimination Practice: Software Tools and Applications. Imperial College Press, London (2003)
18. Boulier, F.: Differential Elimination and Biological Modelling. Radon Series on Computational and Applied Mathematics (Gröbner Bases in Symbolic Analysis) 2, 111–139 (2007), http://hal.archives-ouvertes.fr/hal-00139364
19. Lemaire, F., Moreno Maza, M., Xie, Y.: The RegularChains library in MAPLE 10. In: Kotsireas, I.S. (ed.) The MAPLE conference, pp. 355–368 (2005)
20. Horn, F., Jackson, R.: General mass action kinetics. Archive for Rational Mechanics and Analysis 47, 81–116 (1972)
21. Sedoglavic, A., Ürgüplü, A.: Expanded Lie Point Symmetry (MAPLE package) (2007), http://www.lifl.fr/~sedoglav/Software
22. Hale, J.K., Koçak, H.: Dynamics and Bifurcations. Texts in Applied Mathematics, vol. 3. Springer, New York (1991)
23. Boulier, F., Lazard, D., Ollivier, F., Petitot, M.: Computing representations for radicals of finitely generated differential ideals. Technical report, Université Lille I, LIFL, 59655, Villeneuve d'Ascq, France (1997); Ref. IT306. December 1998 version published in the HDR memoir of Michel Petitot, http://hal.archives-ouvertes.fr/hal-00139061
24. de Jong, H., Ropers, D.: Qualitative Approaches to the Analysis of Genetic Regulatory Networks. In: Szallasi, Z., Stelling, J., Periwal, V. (eds.) System Modeling in Cell Biology: From Concepts to Nuts and Bolts, pp. 125–147. The MIT Press, Cambridge (2006)

On the Computational Power of Biochemistry

Luca Cardelli[1] and Gianluigi Zavattaro[2]

[1] Microsoft Research, Cambridge, UK
[2] Dip. Scienze dell'Informazione, Università di Bologna, Italy

Abstract. We explore the computational power of biochemistry with respect to basic chemistry, identifying complexation as the basic mechanism that distinguishes the former from the latter. We use two process algebras, the Chemical Ground Form (CGF) which is equivalent to basic chemistry, and the Biochemical Ground Form (BGF) which is a minimalistic extension of CGF with primitives for complexation. We characterize an expressiveness gap: CGF is not Turing complete while BGF supports a finite precise encoding of Random Access Machines, a well-known Turing powerful formalism.

1 Introduction

In this paper we introduce a minimal process algebra that aims to capture the essential primitives of biochemistry. Biochemistry is obviously based on chemistry, and in principle one can always express the behavior of a biochemical system by a collection of chemical reactions. But there is a major practical problem with that approach: the collection of reactions for virtually all biochemical systems is an infinite one. For example, just to express the chemical reactions involved in linear polymerization, we need to have a different chemical species for each length n of polymer P_n, with reactions to grow the polymer: $P_n + M \rightarrow P_{n+1}$. While each polymer is finite, the set of possible polymerization reactions is infinite. Nature adopts a more modular solution: the act of joining two molecules is called *complexation*, and polymers are made by iteratively complexing monomers. Each monomer obeys a *finite* simple set of rules that leads to the formation of polymers of any length; therefore, it seems that there should be a finite way of describing such systems. One can start by writing pseudo-reactions like $P + M \rightarrow P : M$, where $P : M$ is meant to represent a P(olymer) molecule attached to an extra M(onomer), yielding a longer polymer. However, there are in general many possible ways (that is, many different patches on the surface of a molecule) by which one molecule can exclusively form a complex with other molecules, and soon one needs to describe the *interface* of each molecule. This situation, while not commonly found in basic chemistry, is particularly acute in biochemistry, where virtually all reactions are governed by enzymes and molecular machines, which are themselves often built by complexation, and which usually operate by complexing with their reactants.

The intuitive idea of a molecule as a stateful entity with a connectivity interface is now common. Notations have emerged from biology that use such

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 65–80, 2008.

an idea to describe large biochemical systems [9,8]. Many formalized and computerized approaches are currently being developed, including: practical tools, where molecules are drawn as boxes with connecting lines [6]; graph-rewriting and term-rewriting systems where a molecular complex is represented as a graph or term, and a reaction is a graph or term rewrite [7,5]; coding techniques in process algebra, where complexation can be expressed via some advanced features [16]; and finally, specialized process algebras where molecular interfaces and complexation are taken as primitive [15,4]. All these approaches aim to find a descriptive framework that goes beyond simple chemical reactions, and that can be used to represent common biochemical situations finitely and modularly.

The aim of this paper is then to investigate the computational boundary between chemistry and biochemistry. That is: what is the intrinsic power of complexation that gives it the ability to represent finitely what would otherwise have an infinite representation? To clarify this issue we study two formal systems, which for easy comparison are both based on the notion of molecules as stateful entities with an interface. One system, the Chemical Ground Form (CGF) has been presented in [3]: it is equivalent to basic chemistry, and it does not include complexation. The other system, the Biochemical Ground Form (BGF) is proposed in this paper as a minimalistic extension of CGF with complexation. As already mentioned, many richer formalisms can represent complexation too, but they also include mechanisms that have no direct biological implementation. Our proposal is minimalistic in the following sense: it adds only two basic actions called *association* and *dissociation*. Association allows two molecules to form a complex, dissociation allows them to subsequently break such a complex. Between an association event involving two molecules and their subsequent dissociation, the two molecules can still freely interact with other molecules or among themselves. In other more expressive formalisms, see for instance the so-called *exchange reactions* in [5], it is possible to specify events that change the internal state of one molecule only if it is complexed with another one having a particular state.

The main contribution of this paper is the formalization of the following expressiveness gap between chemistry and biochemistry: the CGF is not Turing complete while its minimalistic extension BGF is already Turing complete. The results on the CGF are obtained by resorting to papers on the computational power of basic discrete chemistry. In particular, we refer to works by Magnasco [11] and Soloveichik et al. [17] to show that only infinite CGF representations could be sufficiently expressive to precisely model any Turing powerful formalism. On the contrary, we show (as an original result) a finite BGF representation of Random Access Machines [14], a well known register based Turing powerful formalism.

The paper is structured as follows. In Section 2 we give the definition of the CGF and we discuss its computational power. In Section 3 we introduce the BGF, the new notation that enriches the CGF with complexation. In Section 4 we prove that the new process algebra is Turing complete, and finally in Section 5 we give some concluding remarks.

2 Chemical Ground Form

In this section we give the definition of the Chemical Ground Form (CGF): the notation for the representation of chemical systems presented in [3]. We first informally recall the notation, then we give the formal syntax and semantics.

In the CGF each species has an associated definition describing the possible actions for the molecules of that species. Each action $\pi_{(r)}$ has an associated stochastic rate r (a positive real number) which quantifies the expected execution time for the action π.

There are three kinds of actions. Action $\tau_{(r)}$ indicates the possibility for a molecule to be engaged in a unary reaction. For instance, the definition $A = \tau_{(r)}; (B|C)$ is used to specify the possibility for one molecule of species A to be engaged in a unary reaction that produces two molecules, one of species B and one of species C (the operator "|" is borrowed from process algebras such as CCS [13], where it represents parallel composition, and corresponds here to the chemical "+"). Binary reactions have two reactants. The two reactants perform two complementary actions $?a_{(r)}$ and $!a_{(r)}$, where a is a name used to identify the reaction; both the name a and the rate r must match for the reaction to be enabled. For instance, given the definitions $A = ?a_{(r)}; C$ and $B = !a_{(r)}; D$, we have that two molecules of species A and B can be engaged in a binary reaction that produces two molecules, one of species C and one of species D. If the molecules of one species can be engaged in several reactions, then the corresponding definition admits a choice among several actions. The syntax of choice is as follows: $A = \tau_{(r)}; B \oplus ?a_{(r')}; C$, meaning that molecules of species A can be engaged in either a unary reaction that produces a molecule of species B, or in a binary reaction with another molecule able to execute the complementary action $!a_{(r')}$. In the second case, the molecule of species A contributes to the reaction by producing a new molecule of species C.

We now present the formal definition of the syntax of the CGF.

Definition 1 (Chemical Ground Form (CGF)). *Consider the following denumerable sets:* Species *ranged over by variables* $X, Y, \cdots$, Channels *ranged over by* $a, b, \cdots$, *Moreover, let* $r, s, \cdots$ *be rates (i.e. positive real numbers).*

The syntax of CGF is as follows (where the big $\mid$ *separates syntactic alternatives while the small* $|$ *denotes parallel composition):*

$$
\begin{array}{rcll}
E & ::= & \mathbf{0} \mid X = M, E & \textit{Reagents} \\
M & ::= & \mathbf{0} \mid \pi; P \oplus M & \textit{Molecule} \\
P & ::= & \mathbf{0} \mid X|P & \textit{Solution} \\
\pi & ::= & \tau_{(r)} \mid ?a_{(r)} \mid !a_{(r)} & \textit{Internal, Input, Output actions} \\
CGF & ::= & (E, P) & \textit{Reagents and initial Solution}
\end{array}
$$

Given a CGF (E, P)*, we assume that for every variable* X *occurring in* P *or* E*, there is exactly one definition* $X = M$ *in* E*.*

In the following, trailing $\mathbf{0}$ are usually left implicit, and we use $|$ also as an operator over the syntax: if P and P' are $\mathbf{0}$-terminated lists of variables, ac-

cording to the syntax above, then $P|P'$ means appending the two lists into a single $\mathbf{0}$-terminated list. Therefore, if P is a solution, then $\mathbf{0}|P$, $P|\mathbf{0}$, and P are syntactically equal.

We consider the discrete state semantics for the CGF defined in [3] in terms of Continuous Time Markov Chains (CTMCs). The states of the CTMCs are solutions in normal form denoted with $P^\dagger$: for a solution P, we indicate with $P^\dagger$ the normalized form of P where the variables are sorted in lexicographical order (with $\mathbf{0}$ at the end), possibly with repetitions. The CTMC of a chemical ground form is obtained in two steps: we first define the Labeled Transition Graph (LTG) of a chemical ground form, then we show how to extract a CTMC from the labeled transition graph.

In order to define the LTG of a chemical ground form we need to introduce the following notation. Let $E.X$ be the molecule defined by X in E, and $M.i$ be the i-th summand in a molecule of the form $M = \pi_1; P_1 \oplus \cdots \oplus \pi_n; P_n$. Given a solution in normal form $P^\dagger$, with $P^\dagger.m$ we denote the m-th variable in $P^\dagger$, with $P^\dagger \backslash (m_1, \cdots, m_n)$ we denote the solution obtained by removing from $P^\dagger$ the m_i-th molecule for each $i \in \{1, \cdots, n\}$.

A *Labeled Transition Graph* (LTG) is a set of quadruples $\langle l : S^\dagger \xrightarrow{r} T^\dagger \rangle$ where the transition labels l are either of the form $\{m.X.i\}$ or $\{m.X.i, n.Y.j\}$, where m, n, i, j are positive integers, X, Y are species names, $m.X.i$ are ordered triples and $\{\cdots, \cdots\}$ are unordered pairs.

Definition 2 (Labeled Transition Graph (LTG) of a Chemical Ground Form). *Given the Chemical Ground Form (E, P), we define $Next(E, P)$ as the set containing the following kinds of labeled transitions:*

Unary: $\langle \{m.X.i\} : P^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $P^\dagger.m = X$ *and* $E.X.i = \tau_{(r)}; Q$ *and* $T = (P^\dagger \backslash m)|Q$;

Binary: $\langle \{m.X.i, n.Y.j\} : P^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $P^\dagger.m = X$ *and* $P^\dagger.n = Y$ *and* $m \neq n$ *and* $E.X.i = ?a_{(r)}; Q$ *and* $E.Y.j = !a_{(r)}; R$ *and* $T = (P^\dagger \backslash m, n)|Q|R$.

The Labeled Transition Graph of (E, P) is defined as follows:

$$LTG(E, P) = \bigcup_n \Psi_n$$
$$\text{where } \Psi_0 = Next(E, P) \text{ and } \Psi_{n+1} = \bigcup \{Next(E, Q) \mid Q \text{ is a state of } \Psi_n\}$$

We now define how to extract from an LTG the corresponding CTMC.

Definition 3 (Continuous Time Markov Chain of an LTG). *If Ψ is an LTG, then $|\Psi|$ is its CTMC, defined as the set of the triples $P \overset{r}{\mapsto} Q$ with $P \neq Q$, obtained by summing the rates of all the transitions in Ψ that have the same source and target state: $|\Psi| = \{P \overset{r}{\mapsto} Q$ s.t. $\exists \langle l : P \xrightarrow{r'} Q \rangle \in \Psi$ with $P \neq Q$, and $r = \sum r_i$ s.t. $\langle l_i : P \xrightarrow{r_i} Q \rangle \in \Psi\}$.*

We conclude this section by discussing the expressive power of the CGF. First of all, we recall the equivalence result proved in [3] between the CGF and *discrete chemistry*, that is, the traditional stochastic model of chemical kinetics that describes interactions among integer numbers of molecules as CTMCs [12]. More precisely, it is proved that every discrete chemical model (with unary and

binary reactions) has a semantically equivalent CGF, and vice versa. By semantic equivalence between a chemical discrete model and a CGF, we mean that the underlying CTMCs are isomorphic.

In [11], Magnasco shows how to represent in discrete chemistry the computational model of electronic digital computers, based on finite logical circuits with an unbounded memory. Using the translation in [3], we can obtain an equivalent model also in CGF. This is not sufficient to prove that CGF is Turing complete. In fact, the technique proposed by Magnasco requires the exploitation of new species every time a new memory location is needed during the computation. Thus, the CGF system corresponding to a computable function requiring unbounded memory should include an unbounded number of species, thus also an unbounded number of definitions. This is not admitted in the CGF syntax.

The question about Turing completeness of the CGF can be answered in the light of more recent results proved in [17] by Soloveichik et al. In fact, they prove that discrete chemistry is not expressive enough to precisely model any Turing complete formalism, because the problem of deciding whether a certain molecule could be produced in a given chemical system is decidable. Therefore, we can conclude that also the CGF is not Turing complete because, by contraposition, if the CGF were Turing complete, the translation from CGF to discrete chemistry in [3] would allow one to model in discrete chemistry a Turing complete formalism with a finite number of species. It is also possible to derive this result more directly by a connection between the CGF and decidable properties of Petri nets as shown, e.g., in [18].

3 Biochemical Ground Form

In this section we present a minimalistic process algebra for biochemistry, obtained by extending the CGF with association and dissociation. We call the new process algebra *biochemical ground form* (BGF). Following a similar proposal outlined in [2], we consider two additional pairs of complementary actions, $\&?a_{(r)}, \&!a_{(r)}$ for association and $\%?a_{(r)}, \%!a_{(r)}$ for dissociation. Before presenting the formal syntax and semantics of the new actions, we introduce them informally by means of examples. To simplify the notation, in the examples we abstract away from the stochastic rates, e.g., we write $\&?a$ instead of $\&?a_{(r)}$.

Example 1 (Linearly growing polymer). Each complexation event involves exactly two partners. We imagine that the partners have two complementary surface patches that can interlock. If c represents a surface shape (say, a paraboloid), then $!c$ indicates one of the two patches (say, the convex one) and $?c$ indicates the complementary patch (the concave one). Then, $\&!c$ is the action that presents the convex patch, and $\&?c$ is the action that presents the concave patch. When two such *association* actions meet, an actual complexation event can take place, joining the two complementary surfaces.

A linearly growing polymer could be represented as follows, using a seed S and a collection of equal monomers M. The seed starts the chain by presenting a concave patch $?c$: this is our initial, zero-length, polymer. Each monomer

presents a convex patch $!c$, which can bind with an existing polymer on the complementary concave patch. After (and only after) such a binding, a bound monomer M' presents another concave patch $?c$, so that the polymer can keep growing. Both the seed and each monomer can have further behavior, S' and M''.

$$\begin{aligned} S &= \&?c; S' \\ M &= \&!c; M' \\ M' &= \&?c; M'' \end{aligned}$$

Each complexation event creates a unique bond between exactly the two molecules that are joined to each other. This bond needs to be represented somehow, to make sure that a molecule can bind with only one other molecule at a time on any given patch. We represent such a bond as a unique key k that is shared by the two complexed molecules (think of k as a fresh number, or as a fresh channel in π-calculus [13]). Such unique keys, and related information, are collected in the *association history* of each molecule. So, the first interaction of an S with an M, which initially have empty association histories ($\mathbf{0}$), proceeds as follows:

$$S_{\mathbf{0}} \mid M_{\mathbf{0}} \rightarrow S'_{\langle ?c,k1\rangle} \mid M'_{\langle !c,k1\rangle}$$

Interaction with a second monomer then introduces a second fresh key in the histories:

$$S_{\mathbf{0}} \mid M_{\mathbf{0}} \mid M_{\mathbf{0}} \rightarrow S'_{\langle ?c,k1\rangle} \mid M'_{\langle !c,k1\rangle} \mid M_{\mathbf{0}} \rightarrow S'_{\langle ?c,k1\rangle} \mid M''_{\langle ?c,k2\rangle::\langle !c,k1\rangle} \mid M'_{\langle !c,k2\rangle}$$

This mechanism of creation of fresh association keys is repeated every time a new association is created between a monomer and the subsequent one.

It is worth observing that, in any reachable configuration, we can reconstruct from the association histories who is bound to whom, and on what surface the bond was formed. Note that the description of the system is finite (3 reagents, S, M, M'), but that polymers of any length can be assembled (assuming the initial availability of a corresponding amount of monomers).

Example 2 (Branching polymer). After complexation, a molecule is still free to perform additional complexations or other interactions. That is, complexation places no restrictions on the behavior of the original molecules, except for the fact that new complexations cannot occur on surfaces that are already occupied, and that decomplexations must happen consistently with prior complexations (as we discuss shortly). To illustrate this freedom, let us modify the previous example and allow each bound monomer to offer a seed for growing a new polymer branch:

$$\begin{aligned} S &= \&?c; S' \\ M &= \&!c; M' \\ M' &= \&?c; M'' \\ M'' &= \&?d; M''' \\ N &= \&!d; N' \\ N' &= \&?c; N'' \end{aligned}$$

Where an M'' can bind through the interface d to an adaptor molecule N, which then offers another c surface for branching.

Example 3 (Actin-like polymer). Decomplexation is the inverse of complexation, that is, two formerly joined molecules can dissociate. We indicate by $\%!c$ the attempt to dissociate from the convex side, and $\%?c$ the attempt to dissociate from the concave side. When two complexed molecules attempt complementary dissociations, an actual decomplexation event can take place. To illustrate this situation, we describe a different kind of linear polymer: one that can grow only at one end, and can shrink only at the other end. There are four molecular states for each monomer: M^f (free monomer), M^l (monomer bound on the left), M^r (monomer bound on the right), and M^b (monomer bound on both sides). Each monomer has a left convex surface and a complementary right concave surface. A polymer should associate (grow) only on the right and should dissociate (shrink) only on the left.

$$\begin{aligned} M^f &= \&!c; M^l \oplus \&?c; M^r \\ M^l &= \%!c; M^f \oplus \&?c; M^b \\ M^r &= \%?c; M^f \\ M^b &= \%!c; M^r \end{aligned}$$

A free monomer M^f can either associate on the left convex surface and become bound on the left, or associate on the right concave surface and become bound on the right. A monomer M^l bound only on the left can either dissociate on the left (if allowed by its partner, which must in fact be an M^r in this case) and return free, or associate on the right (with an M^f) and become bound on both sides. A monomer M^r bound only on the right can only dissociate on the right: that is, a polymer cannot grow on the left. A monomer M^b bound on both sides can only dissociate on the left (with an M^r): that is, a polymer cannot shrink on the right or break in the middle. These rules cover also the base cases when a polymer of length 2 initially forms or finally dissolves.

A decomplexation should succeed only between a pair of molecules that were actually complexed in their past history, and this can be checked by inspecting the unique keys introduced during complexation. For example let us consider two M^f molecules that complex and then immediately decomplex:

$$M^f_{\mathbf{0}} \mid M^f_{\mathbf{0}} \to M^l_{\langle !c,k\rangle} | M^r_{\langle ?c,k\rangle} \to M^f_{\mathbf{0}} | M^f_{\mathbf{0}}$$

The second transition is allowed to happen because M^l offers $\%!c$, M^r offers the complementary $\%?c$, and the same key k appears in both association histories on the c interface (and with the correct convexity). As a consequence of decomplexation, the keys are removed from the histories.

Example 4 (Unbounded linearly growing and shrinking polymer). Recursive definitions of the species behavior allows us to specify systems in which an unbounded number of monomers can be created. We use this ability to specify a linearly growing polymer started by a seed, that can also shrink removing the last associated monomer, and for which there is no fixed maximal length. In order to produce an unbounded number of monomer we consider a *factory* species able to continuosly produce monomers:

$$\begin{array}{lcl} Fact & = & \tau; (M^f | Fact) \\ S & = & \&?c; S' \\ S' & = & \%?c; S \\ M^f & = & \&!c; M^l \\ M^l & = & \%!c; M^f \oplus \&?c; M^b \\ M^b & = & \%?c; M^l \end{array}$$

It is easy to see that each seed molecule of species S has the ability to start the creation of a polymer that can grow and shrink along one direction without any fixed bound to its maximal length. We will exploit this technique in the proof of Turing completeness of the biochemical ground form in order to model registers, i.e., data structures on which increment, decrement and test for zero operations can be executed. The intuition is that increments are modeled by means of the creation and association of a new monomer, decrements by means of the elimination of the last associated monomer, and test for zero simply by checking the availability of a molecule of species S (the seed becomes of species S' when associated to a monomer).

Almost all new ingredients of the BGF have been presented in the examples above. The unique additional aspect that requires discussion deals with *molecule splitting*, that is the possibility for one reactant to produce more than one molecule. We allow only molecules without complexations (i.e. with an empty association history) to split. In fact, if we admit the splitting of complexed molecules, we also need to extend the language to allow for the specification of the distribution of the associations among the produced molecules: this is possible but somewhat cumbersome. The restriction to splitting only uncomplexed molecules simplifies the notation without limiting the computational power of the calculus.

The complete syntax of the BGF is defined as follows.

Definition 4 (Biochemical Ground Form (BGF)). *Consider the following denumerable sets:* Species *ranged over by variables* X, Y, X^1, X^2, $\cdots$, Channels *ranged over by* a, b, $\cdots$, *a totally ordered set of* Association keys *ranged over by* k, k', $\cdots$. *Moreover, let* r, s, $\cdots$ *be rates (i.e. positive real numbers).*

The syntax of BGF is as follows:

$$\begin{array}{rcll} E & ::= & \mathbf{0} \mid X = M, E & \textit{Reagents} \\ M & ::= & \mathbf{0} \mid \pi; P \oplus M & \textit{Molecule} \\ P & ::= & \mathbf{0} \mid X | P & \textit{Product} \\ \pi & ::= & \tau_{(r)} \mid ?a_{(r)} \mid !a_{(r)} & \textit{Internal, Input, Output actions} \\ & & \mid \&?a_{(r)} \mid \&!a_{(r)} & \textit{Association actions} \\ & & \mid \%?a_{(r)} \mid \%!a_{(r)} & \textit{Dissociation actions} \\ S & ::= & \mathbf{0} \mid X_H | S & \textit{Solution} \\ H & ::= & \mathbf{0} \mid \langle ?a, k\rangle :: H \mid \langle !a, k\rangle :: H & \textit{Association history} \\ BGF & ::= & (E, S) & \textit{Reagents and initial Solution} \end{array}$$

Given a BGF (E, S), we assume that for every variable X occurring in P or E, there is exactly one definition $X = M$ in E. Moreover, we assume that an association key k either does not occur in S or it occurs in exactly two associations $\langle ?a, k\rangle$ and $\langle !a, k\rangle$ (for some channel a) stored in the history of two distinct molecules.[1]

In the following, trailing $\mathbf{0}$ are usually omitted also in association histories: for instance, we denote $\langle ?a, k\rangle :: \mathbf{0}$ simply with $\langle ?a, k\rangle$. Following this simplification, we can use a product P to specify a corresponding solution S, including the same molecules as in P, each of which having an empty association history. Moreover, we consider :: also as an operator over the syntax of association histories: for instance, if H and H' are $\mathbf{0}$-terminated association histories, according to the syntax above, then $H :: H'$ means appending the two lists into a single $\mathbf{0}$-terminated list. Therefore, if H is an association history, then $\mathbf{0} :: H$, $H :: \mathbf{0}$, and H are syntactically equal.

The semantics of BGF is defined, analogously to the semantics of CGF, in terms of a CTMC obtained in two steps, first the definition of a Labeled Transition Graph (LTG), then the extraction of a CTMC from the LTG. The second step is obtained in the same way as described in Definition 3. Thus we simply have to introduce a new definition for the LTG.

Due to the presence of the association histories, we need to introduce a new normal form for solutions.

Definition 5 (Normalized Solution). *For a solution S of a well formed BGF, we indicate with $S^\dagger$ the normalized form of S obtained by*

1. *sorting the molecules first lexicographically according to their species name,*
2. *then sorting the molecules of the same species according to their initial key (i.e. the key of the first association in the history) putting the molecules without an initial key (i.e. with an empty history) before those with an initial key,*
3. *and finally, if there are pairs of molecules of the same species with the same initial key k, put the molecule with association $\langle ?a, k\rangle$ before the molecule with association $\langle !a, k\rangle$.*

Note that normalized solutions are well defined because for each pair of syntactically different molecules X_H and $X'_{H'}$ occurring in a well formed BGF, it defines whether X_H should precede $X'_{H'}$, or the vice versa. In fact, the unique case in which this is not defined is when they are of the same species and they both have an empty association history, thus they are syntactically identical.

On normalized solutions $S^\dagger$, we use the usual notation: $S^\dagger.m$ denotes the m-th molecule in $S^\dagger$, with $S^\dagger \backslash (m_1, \cdots, m_n)$ we denote the solution obtained by removing from $S^\dagger$ the m_i-th molecule for each $i \in \{1, \cdots, n\}$. We use also the following notation on association histories: with $H \setminus \langle ?a, k\rangle$ (resp. $H \setminus \langle !a, k\rangle$) we denote the history obtained by removing from H the association $\langle ?a, k\rangle$ (resp., $\langle !a, k\rangle$).

[1] In BGF, we do not admit self-complexation, i.e., the possibility for one molecule to associate with itself. Still, it is possible for complexed molecules to form cycles; e.g., circular polymers.

We now describe how to produce a Labeled Transition Graph from the Biochemical Ground Form (E, S). As in the previous section, $Next(E, S)$ is a set of quadruples $\langle l : S^\dagger \xrightarrow{r} T^\dagger \rangle$.

Definition 6 (LTG of a BGF). *Given a product P, with $P_{\mathbf{0}}$ we denote the solution obtained adding the empty association history $\mathbf{0}$ to the molecules in P. Given the BGF (E, S), we define $Next(E, S)$ as the set containing the following kinds of labeled transitions:*

Unary: $\langle \{m.X.i\} : S^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $S^\dagger.m = X_H$ *and* $E.X.i = \tau_{(r)}; P$ *and* $T = (S^\dagger \backslash m)|V$ *such that*
- *if* $H = \mathbf{0}$ *then* $V = P_{\mathbf{0}}$,
- *if* $H \neq \mathbf{0}$ *then* $P = X'$ *and* $V = X'_H$;

Binary: $\langle \{m.X.i, n.Y.j\} : S^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $S^\dagger.m = X_H$ *and* $S^\dagger.n = Y_{H'}$ *and* $m \neq n$ *and* $E.X.i = ?a_{(r)}; P$ *and* $E.Y.j = !a_{(r)}; Q$ *and* $T = (S^\dagger \backslash m, n)|V$ *such that*
- *if* $H = \mathbf{0}$ *and* $H' = \mathbf{0}$ *then* $V = P_{\mathbf{0}}|Q_{\mathbf{0}}$,
- *if* $H = \mathbf{0}$ *and* $H' \neq \mathbf{0}$ *then* $Q = Y'$ *and* $V = P_{\mathbf{0}}|Y'_{H'}$,
- *if* $H \neq \mathbf{0}$ *and* $H' = \mathbf{0}$ *then* $P = X'$ *and* $V = X'_H|Q_{\mathbf{0}}$,
- *if* $H \neq \mathbf{0}$ *and* $H' \neq \mathbf{0}$ *then* $P = X'$ *and* $Q = Y'$ *and* $V = X'_H|Y'_{H'}$;

Complexation: $\langle \{m.X.i, n.Y.j\} : S^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $S^\dagger.m = X_H$ *and* $S^\dagger.n = Y_{H'}$ *and* $m \neq n$ *and* $E.X.i = \&?a_{(r)}; X'$ *and* $E.Y.j = \&!a_{(r)}; Y'$ *and for each* k' *we have that* $\langle ?a, k' \rangle \notin H$ *and* $\langle !a, k' \rangle \notin H'$ *and* $T = (S^\dagger \backslash m, n)|X'_{\langle ?a,k \rangle :: H}|Y'_{\langle !a,k \rangle :: H}$ *where* k *is the smallest association key among those that do not appear in the association histories in* $S^\dagger$;

Decomplexation: $\langle \{m.X.i, n.Y.j\} : S^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $S^\dagger.m = X_H$ *and* $S^\dagger.n = Y_{H'}$ *and* $m \neq n$ *and* $E.X.i = \%?a_{(r)}; X'$ *and* $E.Y.j = \%!a_{(r)}; Y'$ *and there exists* k *s.t.* $\langle ?a, k \rangle \in H$ *and* $\langle !a, k \rangle \in H'$ *and* $T = (S^\dagger \backslash m, n)|X'_{H \backslash \langle ?a,k \rangle}|Y'_{H' \backslash \langle !a,k \rangle}$.

The Labeled Transition Graph of (E, S) is defined as follows:

$$LTG(E, S) = \bigcup_n \Psi_n$$
where $\Psi_0 = Next(E, S)$ *and* $\Psi_{n+1} = \bigcup \{Next(E, Q) \mid Q \text{ is a state of } \Psi_n\}$

It is easy to see that the assumption at the end of the Definition 4, i.e. that an association key k either does not occur in the solution or it occurs in exactly two associations $\langle ?a, k \rangle$ and $\langle !a, k \rangle$ (for some channel a) stored in the history of two distinct molecules, is preserved by the labeled transition system. In fact, the unique rules able to modify the association histories (the last two items in the Definition 6), removes both instances of an association key or create two instances of a new key, respectively.

The restriction, that we already informally discussed, that complexed molecules cannot split follows from the fact that splitting is possible only in the first two items of the Definition 6, and only in case the association history of the splitting molecule is empty.

Finally, we define the semantics of a BGF (E, S) as $|LTG(E, S)|$, that is the CTMC obtained from the labeled transition graph $LTG(E, S)$ according to the technique presented in Definition 3.

4 Turing Completeness of BGF

We prove that Biochemical Ground Form is Turing complete. This result allows us to conclude that the association and dissociation actions cannot be encoded in the CGF, because the addition of these mechanisms makes BGF strictly more expressive.

In order to prove that BGF is Turing complete, we show how to model Random Access Machines (RAMs) [14], a well known Turing powerful formalism based on registers containing nonnegative natural numbers. The registers are used by a program, that is a set of indexed instructions I_i of two possible kinds:

- $i : Inc(r_j)$ that increments the register r_j and then moves to the execution of the instruction with index $i + 1$ and
- $i : DecJump(r_j, s)$ that attempts to decrement the register r_j; if the register does not hold 0 then the register is actually decremented and the next instruction is the one with index $i + 1$, otherwise the next instruction is the one with index s.

We assume the existence of a special instruction I_{halt} corresponding to program termination.

In our encoding of RAMs, we use a simplified notation for BGF definitions in which actions can be written in sequence. For instance the definition $A = \pi_1; \pi_2; C$ is a shorthand for the two definitions $A = \pi_1; B$ and $B = \pi_2; C$. Moreover, we do not show the stochastic rates (r) of the actions as they are not relevant: the execution of a RAM encoding proceeds deterministically (there are no probabilistic choices governed by the rates) and the speed of the RAM simulation is not important.

The encoding considers one species I^i for each instruction I_i. The behavior of the molecules of species I^i is to update the registers according to the corresponding instruction I_i, and then produce one molecule of species I^j corresponding to the subsequent instruction to be executed.

Formally, the species corresponding to the instructions are defined as follows:

$$I^i = \begin{cases} !inc_j; ?ack; I^{i+1} & \text{if } I_i = i : Inc(r_j) \\ !dec_j; ?ack; I^{i+1} \oplus !zero_j; I^s & \text{if } I_i = i : DecJump(r_j, s) \\ \mathbf{0} & \text{if } I_i = I_{halt} \end{cases}$$

In Figure 1 we graphically depict the above definitions using a graph-like notation: each species is represented by one node, and a transition labeled with an action π represents the possibility for the molecules of the source species to perform the action π producing one molecule of the target species. In case of an increment instruction, a request for increment inc_j is considered, then an acknowledgment is required to have confirmation that the increment actually took place, and finally the next instruction is activated. In case of a decrement, either a decrement or a test for emptiness can take place: in the first case an acknowledgment is required before activating the next instruction; in the second case the jump is executed. In case of the terminating instruction, the corresponding molecule simply does nothing.

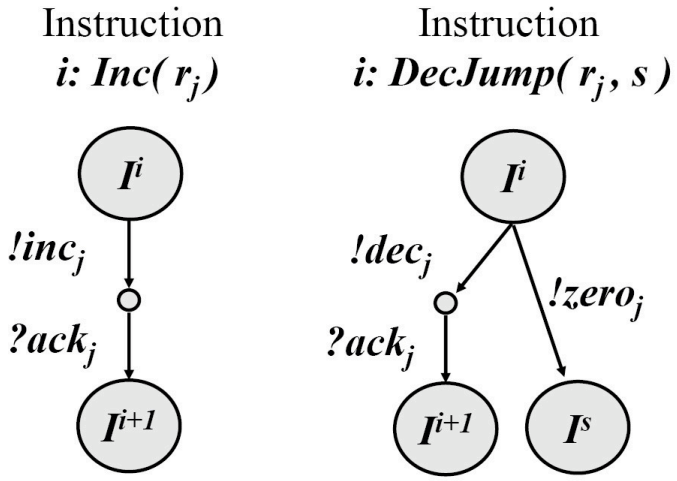


Fig. 1. Encoding of RAM instructions

Each register r_j is modeled by a polymer similar to those described in the Example 4. In this case the seed is of species Z^j and the monomers are of species R^j. The number of monomers in the polymer coincides with the register content, namely, when the register holds the number l the polymer is composed of exactly l monomers. As it is not possible to know a priori the number of monomers necessary during the computation, we consider a factory, that is, a molecule of species RF^j which is responsible for the the generation of the molecules of species R^j whenever they are needed. The last associated molecule in the polymer is the only one able to interact with the instruction molecules: if it is of species Z^j the active action is $?zero_j$, if it is of species R^j the active action is $?dec_j$. The effect of the execution of $?dec_j$ is the dissociation of the last associated molecule from the polymer.

The formal definition of the species used to model registers is as follows:

$$\begin{array}{lcl} Z^j & = & ?zero_j; Z^j \ \oplus \ \&?link_j; \%?link_j; Z^j \\ RF^j & = & ?inc_j; \left(RF^j | (\&!link_j; !ack; R^j)\right) \\ R^j & = & (\&?link_j; \%?link_j; R^j_{!link_j}) \ \oplus \ (?dec_j; \%!link_j; !ack; \mathbf{0}) \end{array}$$

In Figure 2 we graphically depict the encoding of registers. In this case we also have to represent the splitting of the molecules of species RF^j when performing the action $?inc_j$: the transition enters an intermediary *splitting* state represented with a bar, from which we have one outgoing transition for each of the produced molecules.

The remainder of this section is devoted to the formal proof of correctness of this RAM encoding. We use the following notation. Given a RAM with registers $r_1, \cdots, r_n$, we denote with $(I_i, r_1 = l_1, \cdots, r_n = l_n) \mapsto (I_j, r_1 = l'_1, \cdots, r_n = l'_n)$ its possible steps of computation. Namely, if the RAM is going to execute instruction I_i, and the register contents are $l_1, \cdots, l_n$, respectively, then the next instruction is I_j and the new register contents are $l'_1, \cdots, l'_n$, respectively.

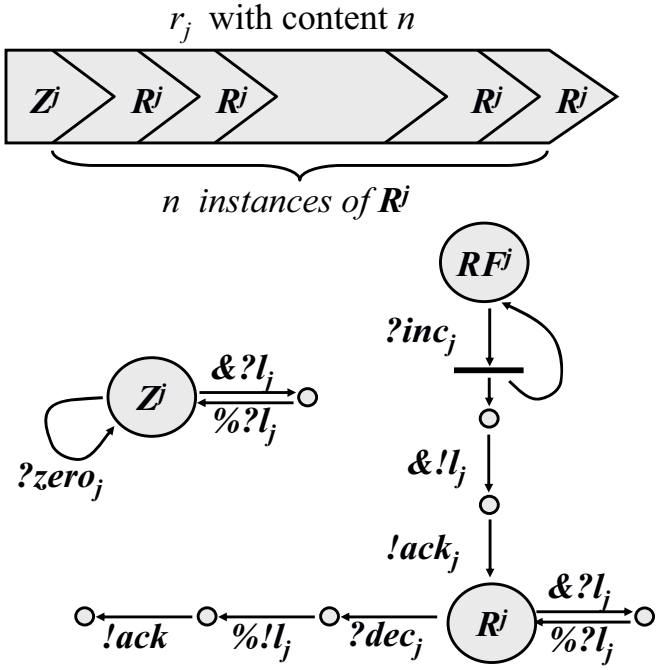


Fig. 2. Encoding of RAM registers

In the following we need to treat as equivalent some syntactically different solutions which represent the same biological system. For instance, the two solutions

$$Z^1_{\langle ?link_1,1\rangle} \mid R^1_{\langle ?link_1,4\rangle::\langle !link_1,1\rangle} \mid R^1_{\langle !link_1,4\rangle}$$
$$Z^1_{\langle ?link_1,2\rangle} \mid R^1_{\langle ?link_1,3\rangle::\langle !link_1,2\rangle} \mid R^1_{\langle !link_1,3\rangle}$$

both denote the polymer representing the register r_1 with content 2, even if they differ in their association keys. Formally, we have that two solutions S and T are *equivalent* if there exists an injective renaming ρ for the association keys in S such that $(S[\rho])^\dagger = T^\dagger$, where $S[\rho]$ denotes the result of the application of the injective renaming to the solution S. In the example above, the injective renaming used to prove that the two solutions are equivalent is $\{1 \mapsto 2, 4 \mapsto 3\}$.

We denote with $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$ the set of equivalent solutions which represent the RAM ready to execute the instruction I_i and in which the registers $r_1, \cdots, r_m$ have contents $l_1, \cdots, l_m$, respectively. Formally, $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$ is the set of solutions equivalent to:

$$\begin{array}{l} I^i \mid \\ RF^1 \mid Z^1_{\langle ?link_1,k^1_1\rangle} \mid R^1_{\langle ?link_1,k^2_1\rangle::\langle !link_1,k^1_1\rangle} \mid \cdots \mid R^1_{\langle !link_1,k^{l_1}_1\rangle} \mid \\ \cdots \\ RF^m \mid Z^m_{\langle ?link_m,k^1_m\rangle} \mid R^m_{\langle ?link_m,k^2_m\rangle::\langle !link_m,k^1_m\rangle} \mid \cdots \mid R^m_{\langle !link_m,k^{l_m}_m\rangle} \end{array}$$

Given a RAM denoted with $\mathcal{R}$, having instructions $I_1, \cdots, I_n$ and registers $r_1, \cdots, r_m$, we use $E_\mathcal{R}$ to denote the definitions of the species I^1, $\cdots$, I^n, Z^1, $\cdots$, Z_m, RF^1, $\cdots$, RF^m, and R^1, $\cdots$, R^m as defined above. Thus, given one of

the possible configurations $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ of $\mathcal{R}$, we model it with the BGF $(E_{\mathcal{R}}, S)$ where S is any of the solutions in $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$.

We are now ready to prove the correctness result.

Theorem 1. *Let $\mathcal{R}$ be a RAM. Given one of its possible configurations $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ and a solution $S_0 \in [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$, we have that:*

- *either $I_i = I_{halt}$ and $Next(E_{\mathcal{R}}, S)$ is empty;*
- *or $(I_i, r_1 = l_1, \cdots, r_m = l_m) \mapsto (I_j, r_1 = l'_1, \cdots, r_m = l'_m)$ and there exist $S_1, \cdots, S_z$ such that for every $0 \leq x < z$ we have that $Next(E_{\mathcal{R}}, S^{\dagger}_x)$ contains only one transition which has $S^{\dagger}_{x+1}$ as its target state, and moreover $S_z \in [\![(I_j, r_1 = l'_1, \cdots, r_m = l'_m)]\!]$.*

Proof (outline). The proof is by case analysis on the following four possible cases: $I_i = I_{halt}$, $I_i = i : Inc(r_j)$, $I_i = i : DecJump(r_j, s)$ with $l_j > 0$, and $I_i = i : DecJump(r_j, s)$ with $l_j = 0$.

As a corollary of the theorem above, we have that if $\mathcal{R}$ is a RAM, given one of its possible configurations $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ and a solution $S \in [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$, there exists a solution containing the molecule I^{halt} in $|LTG(E_{\mathcal{R}}, S)|$ if and only if the computation starting from the configuration $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ halts. As the halting problem is undecidable for RAMs, we have that in the BGF the problem of deciding whether a certain molecule could be produced in a given system is undecidable. We have already observed that, on the contrary, this property is decidable for the process algebra CGF.

5 Conclusion

Turing-powerful mechanisms are not a requirement for building sophisticated nano-machines. Yet, the existence of Turing-powerful mechanisms guarantees a certain level of generality and flexibility in constructing machinery of any desired complexity, and provides evolution with adaptable toolkits to build upon. This paper highlights the fact that nature widely employs Turing-powerful mechanisms at the molecular level, and that it does so in a finitary combinatorial way that is qualitatively different from the common notion of chemical reactions between simple species. We have shown that the biochemical operations of complexation and decomplexation, formalized in a very basic form, are sufficient to raise expressiveness to the level of Turing-completeness, while simple chemistry (with finite descriptions) is not sufficient. In other words, finite programming constructs that are Turing powerful can be found in biochemistry but not in simple chemistry.

It is interesting to note that similar computational boundaries have been proved also in the context of process calculi based on membrane interactions such as endocytosis, exocytosis, fusion, and fission. In [1], Busi and Gorrieri prove that a basic process calculus including endoctytosis and exocytosis is Turing complete, while this is not the case when only fusion and fission are considered.

This because endocytosis allows for the nesting of membranes with an unbounded depth, while this is not possible when only fission and fusion are considered. In BGF, instead of using membrane nesting, we consider a more basic complexation mechanism in order to generate structures with unbounded length.

The boundary of Turing-completeness gets even more interesting at the quantitative, approximate, level. For instance, recent work by Liekens and Fernando [10] shows how to approximate in discrete chemistry finite computations of Register Machines with an error probability smaller than any given precision $\delta > 0$. Soloveichik et al. [17], besides proving that in discrete chemistry it is not possible to precisely model any Turing powerful formalism (the result we have used in Section 2 to motivate that CGF is not Turing complete), show also how to approximate unbounded computations. A consequence of their results is that it is always decidable whether a certain molecule *could* be produced in a chemical system, while the question whether the system is *likely* to produce that molecule is in general undecidable. This opens interesting questions about what is actually decidable and what is undecidable in discrete chemistry. Some results recently proved along this line of research can be found in [18].

References

1. Busi, N., Gorrieri, R.: On the Computational Power of Brane Calculi. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 16–43. Springer, Heidelberg (2006)
2. Cardelli, L.: Artificial Biochemistry. In: Proc. of Algorithmic Bioprocesses. LNCS (to appear, 2008), `http://lucacardelli.name`
3. Cardelli, L.: On Process Rate Semantics. Theoretical Computer Science (in press, 2008), `http://dx.doi.org/10.1016/j.tcs.2007.11.012`
4. Cardelli, L., Pradalier, S.: Where Membranes Meet Complexes. In: Proc. of Concurrent Models in Molecular Biology (BioConcur. 2005) (2005)
5. Credi, A., Garavelli, M., Laneve, C., Pradalier, S., Silvi, S., Zavattaro, G.: Modelization and Simulation of Nano Devices in nano-kappa Calculus. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 168–183. Springer, Heidelberg (2007)
6. Danos, V., Feret, J., Fontana, W., Krivine, J.: Kappa Factory (2007), `http://www.lix.polytechnique.fr/~krivine/kappaFactory.html`
7. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science 325(1), 69–110 (2004)
8. Kitano, H., Funahashi, A., Matsuoka, Y., Oda, K.: Using process diagrams for the graphical representation of biological networks. Nature Biotechnolgy 23, 961–966 (2005)
9. Kohn, K.W., Aladjem, M.I., Weinstein, J.N., Pommier, Y.: Molecular interaction maps of bioregulatory networks: a general rubric for systems biology. Molecular biology of the cell 17(1), 1–13 (2006)
10. Liekens, A.M.L., Fernando, C.T.: Turing complete catalytic particle computers. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 1202–1211. Springer, Heidelberg (2007)
11. Magnasco, M.O.: Chemical Kinetics is Turing Universal. Physical Review Letters 78, 1190–1193 (1997)

12. McQuarrie, D.A.: Stochastic approach to chemical kinetics. Journal of Applied Probability 4, 413–478 (1967)
13. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
14. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Englewood Cliffs (1967)
15. Priami, C., Quaglia, P.: Beta Binders for Biological Interactions. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 20–33. Springer, Heidelberg (2005)
16. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters 80, 25–31 (2001)
17. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with Finite Stochastic Chemical Reaction Networks. Natural Computing (in press, 2008), http://dx.doi.org/10.1007/s11047-008-9067-y
18. Zavattaro, G., Cardelli, L.: Termination Problems in Chemical Kinetics (2008), http://lucacardelli.name

The Geometry of the Neighbor-Joining Algorithm for Small Trees

Kord Eickmeyer[1] and Ruriko Yoshida[2]

[1] Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany
[2] University of Kentucky, Lexington, KY, USA

Abstract. In 2007, Eickmeyer et al. showed that the tree topologies outputted by the Neighbor-Joining (NJ) algorithm and the balanced minimum evolution (BME) method for phylogenetic reconstruction are each determined by a polyhedral subdivision of the space of dissimilarity maps $\mathbb{R}^{\binom{n}{2}}$, where n is the number of taxa. In this paper, we will analyze the behavior of the Neighbor-Joining algorithm on five and six taxa and study the geometry and combinatorics of the polyhedral subdivision of the space of dissimilarity maps for six taxa as well as hyperplane representations of each polyhedral subdivision. We also study simulations for one of the questions stated by Eickmeyer et al., that is, the robustness of the NJ algorithm to small perturbations of tree metrics, with tree models which are known to be hard to be reconstructed via the NJ algorithm.

1 Introduction

The Neighbor-Joining (NJ) algorithm was introduced by Saitou and Nei [14] and is widely used to reconstruct a phylogenetic tree from an alignment of DNA sequences because of its accuracy and computational speed. The NJ algorithm is a distance-based method which takes all pairwise distances computed from the data as its input, and outputs a tree topology which realizes these pairwise distances, if there is such a topology (see Fig. 1). Note that the NJ algorithm is consistent, i.e., it returns the additive tree if the input distance matrix is a tree metric. If the input distance matrix is not a tree metric, then the NJ algorithm returns a tree topology which induces a tree metric that is "close" to the input. Since it is one of the most popular methods for reconstructing a tree among biologists, it is important to study how the NJ algorithm works.

	a	**b**	**c**	**d**
a	-	3	1.8	2.5
b		-	2.8	3.5
c			-	1.3
d				-

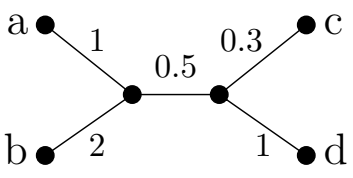


Fig. 1. The NJ algorithm takes a matrix of pairwise distances (left) as input and computes a binary tree (right). If there is a tree such that the distance matrix can be obtained by taking the length of the unique path between two nodes, NJ outputs that tree.

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 81–95, 2008.

A number of attempts have been made to understand the good results obtained with the NJ algorithm, especially given the problems with the inference procedures used for estimating pairwise distances. For example, Bryant showed that the *Q-criterion* (defined in (Q) in Section 2.2) is in fact the unique selection criterion which is linear, permutation invariant, and *consistent*, i.e., it correctly finds the tree corresponding to a tree metric [2]. Gascuel and Steel gave a nice review of how the NJ algorithm works [15].

One of the most important questions in studying the behavior of the NJ algorithm is to analyze its performance with pairwise distances that are not tree metrics, especially when all pairwise distances are estimated via the maximum likelihood estimation (MLE). In 1999, Atteson showed that if the distance estimates are at most half the minimal edge length of the tree away from their true value then the NJ algorithm will reconstruct the correct tree [1]. However, Levy et al. noted that Atteson's criterion frequently fails to be satisfied even though the NJ algorithm returns the true tree topology [10]. Recent work of [11] extended Atteson's work. Mihaescu et al. showed that the NJ algorithm returns the true tree topology when it works locally for the quartets in the tree [11]. This result gives another criterion for when then NJ algorithm returns the correct tree topology and Atteson's theorem is a special case of Mihaescu et al.'s theorem.

For every input distance matrix, the NJ algorithm returns a certain tree topology. It may happen that the minimum Q-criterion is taken by more than one pair of taxa at some step. In practice, the NJ algorithm will then have to choose one cherry in order to return a definite result, but for our analysis we assume that in those cases the NJ algorithm will return a set containing all tree topologies resulting from picking optimal cherries. There are only finitely many tree topologies, and for every topology t we get a subset D_t of the sample space (input space) such that for all distance matrices in D_t one possible answer of the NJ algorithm is t. We aim at describing these sets D_t and the relation between them. One notices that the Q-criteria are all linear in the pairwise distances. The NJ algorithm will pick cherries in a particular order and output a particular tree t if and only if the pairwise distances satisfy a system of linear inequalities, whose solution set forms a polyhedral cone in $\mathbb{R}^{\binom{n}{2}}$. Eickmeyer et al. called such a cone a *Neighbor-Joining cone,* or *NJ cone*. Thus the NJ algorithm will output a particular tree t if and only if the distance data lies in a union of NJ cones [3].

In [3], Eickmeyer et al. studied the optimality of the NJ algorithm compared to the balanced minimum evolution (BME) method and focused on polyhedral subdivisions of the space of dissimilarity maps for the BME method and the NJ algorithm. Eickmeyer et al. also studied the geometry and combinatorics of the NJ cones for $n = 5$ in addition to the *BME cones* for $n \leq 8$. Using geometry of the NJ cones for $n = 5$, they showed that the polyhedral subdivision of the space of dissimilarity maps with the NJ algorithm does not form a fan for $n \geq 5$ and that the union of the NJ cones for a particular tree topology is not convex. This means that the NJ algorithm is not convex, i.e., there are distance matrices D, D', such that NJ produces the same tree t_1 on both inputs D and D', but it produces a different tree $t_2 \neq t_1$ on the input $(D+D')/2$ (see [3] for an example).

In this paper, we focus on describing geometry and combinatorics of the NJ cones for six taxa as well as some simulation study using the NJ cones for one of the questions in [3], that is, what is the robustness of the NJ algorithm to small perturbations of tree metrics for $n = 5$. This paper is organized as follows: In Section 2 we will describe the NJ algorithm and define the NJ cones. Section 3 states the hyperplane representations of the NJ cones for $n = 5$. Section 4 describes in summary the geometry and combinatorics of the NJ cones for $n = 6$. Section 5 shows some simulation studies on the robustness of the NJ algorithm to small perturbations of tree metrics for $n = 5$ with the tree models from [13]. We end by discussing some open problems in Section 6.

2 The Neighbor-Joining Algorithm

2.1 Input Data

The NJ algorithm is a *distance-based method* which takes a *distance matrix*, a symmetric matrix $(d_{ij})_{0\leq i,j\leq n-1}$ with $d_{ii} = 0$ representing pairwise distances of a set of n taxa $\{0, 1, \ldots, n-1\}$, as the input. Through this paper, we do not assume anything on an input data except it is symmetric and $d_{ii} = 0$. Because of symmetry, the input can be seen as a vector of dimension $m := \binom{n}{2} = \frac{1}{2}n(n-1)$. We arrange the entries row-wise. We denote row/column-indices by pairs of letters such as a, b, c, d, while denoting single indices into the "flattened" vector by letters $i, j, \ldots$. The two indexing methods are used simultaneously in the hope that no confusion will arise. Thus, in the four taxa example we have $d_{0,1} = d_{1,0} = d_0$. In general, we get $d_i = d_{a,b} = d_{b,a}$ with

$$a = \max\left\{k \mid \frac{1}{2}k(k-1) \leq i\right\} = \left\lfloor \frac{1}{2} + \sqrt{\frac{1}{4} + 2i} \right\rfloor, b = i - \frac{1}{2}(a-1)a,$$

and for $c > d$ we get

$$d_{c,d} = d_{c(c-1)/2+d}.$$

2.2 The Q-Criterion

The NJ algorithm starts by computing the so called *Q-criterion* or the *cherry picking criterion*, given by the formula

$$q_{a,b} := (n-2)d_{a,b} - \sum_{k=0}^{n-1} d_{a,k} - \sum_{k=0}^{n-1} d_{k,b}. \tag{Q}$$

This is a key of the NJ algorithm to choose which pair of taxa is a neighbor.

Theorem 1 (Saitou and Nei, 1987, Studier and Keppler, 1988 [14,16]). *Let $d_{a,b}$ for all pair of taxa $\{a, b\}$ be the tree metric corresponding to the tree T. Then the pair $\{x, y\}$ which minimizes $q_{a,b}$ for all pair of taxa $\{a, b\}$ forms a neighbor.*

Arranging the Q-criteria for all pairs in a matrix yields again a symmetric matrix, and ignoring the diagonal entries we can see it as a vector of dimension m just like the input data. Moreover, the Q-criterion is obtained from the input data by a linear transformation:

$$\mathbf{q} = A^{(n)}\mathbf{d},$$

and the entries of the matrix $A^{(n)}$ are given by

$$A^{(n)}_{ij} = A^{(n)}_{ab,cd} = \begin{cases} n-4 & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } \{a,b\} \cap \{c,d\} \neq \emptyset, \\ 0 & \text{else,} \end{cases} \tag{1}$$

where $a > b$ is the row/column-index equivalent to i and likewise for $c > d$ and j. When no confusion arises about the number of taxa, we abbreviate $A^{(n)}$ to A.

After computing the Q-criterion $\mathbf{q}$, the NJ algorithm proceeds by finding the minimum entry of $\mathbf{q}$, or, equivalently, the maximum entry of $-\mathbf{q}$. The two nodes forming the chosen pair (there may be several pairs with minimal Q-criterion) are then joined ("cherry picking"), i.e., they are removed from the set of nodes and a new node is created. Suppose out of our n taxa $\{0, \ldots, n-1\}$, the first cherry to be picked is $m-1$, so the taxa $n-2$ and $n-1$ are joined to form a new node, which we view as the new node number $n-2$. The reduced pairwise distance matrix is one row and one column shorter than the original one, and by our choice of which cherry we picked, only the entries in the rightmost column and bottom row differ from the original ones. Explicitly,

$$d'_i = \begin{cases} d_i & \text{for } 0 \leq i < \binom{n-2}{2} \\ \frac{1}{2}(d_i + d_{i+(n-2)} - d_{m-1}) & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2} \end{cases}$$

and we see that the reduced distance matrix depends linearly on the original one:

$$\mathbf{d}' = R\mathbf{d},$$

with $R = (r_{ij}) \in \mathbb{R}^{(m-n+1)\times m}$, where

$$r_{ij} = \begin{cases} 1 & \text{for } 0 \leq i = j < \binom{n-2}{2} \\ 1/2 & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2}, j = i \\ 1/2 & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2}, j = i+n-2 \\ -1/2 & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2}, j = m-1 \\ 0 & \text{else.} \end{cases}$$

The process of picking cherries is repeated until there are only three taxa left, which are then joined to a single new node.

We note that since new distances d' are always linear combinations of the previous distances, all Q-criteria computed throughout the NJ algorithm are linear combinations of the original pairwise distances. Thus, for every possible tree topology t outputted by the NJ algorithm (and every possible ordering σ of

picked cherries that results in topology t), there is a polyhedral cone $C_{T,\sigma} \subset \mathbb{R}^{\binom{n}{2}}$ of dissimilarity maps. The NJ algorithm will output t and pick cherries in the order σ iff the input lies in the cone $C_{T,\sigma}$. We call the cones $C_{T,\sigma}$ *Neighbor-Joining cones*, or *NJ cones*.

2.3 The Shifting Lemma

We first note that there is an n-dimensional linear subspace of $\mathbb{R}^m$ which does not affect the outcome of the NJ algorithm (see [11]). For a node a we define its *shift vector* $\mathbf{s}_a$ by

$$(\mathbf{s}_a)_{b,c} := \begin{cases} 1 & \text{if } a \in \{b, c\} \\ 0 & \text{else} \end{cases}$$

which represents a tree where the leaf a has distance 1 from all other leaves and all other distances are zero. The Q-criterion of any such vector is -2 for all pairs, so adding any linear combination of shift vectors to an input vector does not change the relative values of the Q-criteria. Also, regardless of which pair of nodes we join, the reduced distance matrix of a shift vector is again a shift vector (of lower dimension), whose Q-criterion will also be constant. Thus, for any input vector $\mathbf{d}$, the behavior of the NJ algorithm on $\mathbf{d}$ will be the same as on $\mathbf{d} + \mathbf{s}$ if $\mathbf{s}$ is any linear combination of shift vectors. We call the subspace generated by shift vectors S.

We note that the difference of any two shift vectors is in the kernel of A, and the sum of all shift vectors is the constant vector with all entries equal to n. If we fix a node a then the set

$$\{\mathbf{s}_a - \mathbf{s}_b \mid b \neq a\}$$

is linearly independent.

2.4 The First Step in Cherry Picking

After computing the Q-criterion $\mathbf{q}$, the NJ algorithm proceeds by finding the minimum entry of it, or, equivalently, the maximum entry of $-\mathbf{q}$. The set $cq_i \subset \mathbb{R}^m$ of all q-vectors for which q_i is minimal is given by the normal cone at the vertex $-e_i$ to the (negative) simplex

$$\Delta_{m-1} = \operatorname{conv}\{-e_i \mid 0 \leq i \leq m-1\} \subset \mathbb{R}^m,$$

where $e_0, \ldots, e_{m-1}$ are the unit vectors in $\mathbb{R}^m$. The normal cone is defined in the usual way by

$$\begin{aligned} \mathcal{N}_{\Delta_{m-1}}(-e_i) &:= \{\mathbf{x} \in \mathbb{R}^m \mid (-e_i, \mathbf{x}) \geq (\mathbf{p}, \mathbf{x}) \text{ for } \mathbf{p} \in \Delta_{m-1}\} \\ &= \{\mathbf{x} \in \mathbb{R}^m \mid (-e_i, \mathbf{x}) \geq (-e_j, \mathbf{x}) \text{ for } 0 \leq j \leq m-1\}, \end{aligned} \tag{2}$$

with $(\cdot, \cdot)$ denoting the inner product in $\mathbb{R}^m$.

Substituting $\mathbf{q} = A\mathbf{d}$ into (2) gives

$$\begin{aligned} \mathbf{q} \in cq_i \quad &\Leftrightarrow \quad i = \arg\max(-e_j, A\mathbf{d}) \\ &\Leftrightarrow \quad i = \arg\max(-A^T e_j, \mathbf{d}) \\ &\Leftrightarrow \quad i = \arg\max(-Ae_j, \mathbf{d}) \qquad \text{because } A \text{ is symmetric.} \end{aligned} \tag{3}$$

Therefore the set cd_i of all *parameter* vectors $\mathbf{d}$ for which the NJ algorithm will select cherry i in the first step is the normal cone at $-Ae_i$ to the polytope

$$P_n := \operatorname{conv}\{-Ae_0, \ldots, -Ae_{m-1}\}. \tag{4}$$

The shifting lemma implies that the affine dimension of the polytope P_n is at most $m - n$. Computations using `polymake` show that this upper bound gives the true value.

If equality holds for one of the inner products in this formula, then there are two cherries with the same Q-criterion.

As the number of taxa increases, the resulting polytope gets more complicated very quickly. By symmetry, the number of facets adjacent to a vertex is the same for every vertex, but this number grows following a strange pattern. See Table 1 for some calculated values. We also computed f-vectors for P_n via `polymake`. With $n = 5$, we have $(1, 10, 45, 90, 75, 22, 1)$, with $n = 6$, $(1, 15, 105, 435, 1095, 1657, 1470, 735, 195, 25, 1)$, and with $n \geq 7$ we ran `polymake` over several hours and it took more than 9GB RAM. Therefore, we could not compute them.

Table 1. The polytopes P_n for some small numbers of taxa n

no. of taxa	no. of vertices	dimension	facets through vertex	no. of facets
4	3	2	2	3
5	10	5	12	22
6	15	9	18	25
7	21	14	500	717
8	28	20	780	1,057
9	36	27	30,114	39,196
10	45	35	77,924	98,829

2.5 The Cone cd_i

Equation (3) allows us to write cd_i as an intersection of half-spaces as follows:

$$\begin{aligned} cd_i &= \{\mathbf{x} \mid (-A\mathbf{e}_i, \mathbf{x}) \geq (-A\mathbf{e}_j, \mathbf{x}) \text{ for } j \neq i\} \\ &= \{\mathbf{x} \mid (-A(\mathbf{e}_i - \mathbf{e}_j), \mathbf{x}) \geq 0 \text{ for } j \neq i\} \\ &= \bigcap_{j \neq i} \{\mathbf{x} \mid (-A(\mathbf{e}_i - \mathbf{e}_j), \mathbf{x}) \geq 0\} \end{aligned} \tag{5}$$

We name the half-spaces, their interiors and the normal vectors defining them as follows:

$$\begin{aligned} \mathbf{h}_{ij}^{(n)} &:= -A^{(n)}(\mathbf{e}_i - \mathbf{e}_j), \\ H_{ij}^{(n)} &:= \left\{ \mathbf{x} \in \mathbb{R}^m \mid (\mathbf{h}_{ij}^{(n)}, \mathbf{x}) \geq 0 \right\}, \\ \mathring{H}_{ij}^{(n)} &:= \left\{ \mathbf{x} \in \mathbb{R}^m \mid (\mathbf{h}_{ij}^{(n)}, \mathbf{x}) > 0 \right\}, \end{aligned} \tag{6}$$

where again we omit the superscript (n) if the number of taxa is clear.

If there are more than four taxa, then this representation is not redundant: For any pair i and j of cherries, we can find a parameter vector $\mathbf{d}$ lying on the border of H_{ij} (i.e., $(\mathbf{h}_{ij}, \mathbf{d}) = 0$) but in the interior $\mathring{H}_{ik}$ of the other half-spaces for $k \neq i, j$. One such $\mathbf{d}$ is given by

$$d_k := \begin{cases} 2 & \text{if } k = i \text{ or } k = j, \\ 4 & \text{else.} \end{cases}$$

Thus we have found an $\mathcal{H}$-representation of the polyhedron cd_i consisting of only $m-1$ inequalities. Note that Table 1 implies that a $\mathcal{V}$-representation of the same cone would be much more complicated, as the number of vectors spanning it is equal to the number of facets incident at the vertex $-A\mathbf{e}_i$.

Example 1. The normal vectors to the 22 facets of P_5, and thus the rays of the normal cones to P_5, form two classes (see Fig. 2). The first class contains a total of 12 vectors (as there are 12 assignments of nodes 0 to 4 to the labels a to e which yield nonisomorphic labelings), and every normal cone contains six of them. The second class contains 10 vectors, and again every normal cone has six of them as rays. For each class there are two diagrams in Fig. 2, and we obtain a normal vector to one of the facets of P_5 by assigning nodes from $\{0, \ldots, 4\}$ to the labels $a, \ldots, e$. The left diagram tells which vertices of P_5 belong to the facet defined by that normal vector: Two nodes in the diagram are connected by an edge iff the vertex belonging to that pair of nodes is in the facet. The edges in the right diagram are labeled with the distance between the corresponding pair of nodes in the normal vector. This calls for an example: Setting $a = 0, \ldots, e = 4$, Fig. 2(a) gives a distance vector

$$(d_{01}, d_{02}, \ldots, d_{24}, d_{34})^{\mathrm{T}} = (-1, 1, 1, -1, -1, 1, 1, -1, 1, -1)^{\mathrm{T}},$$

which is a common ray of the cones cd_{01}, cd_{12}, cd_{23}, cd_{34} and cd_{04}. Thus of the 22 facets of P_5, 12 have five vertices and 10 have six vertices.

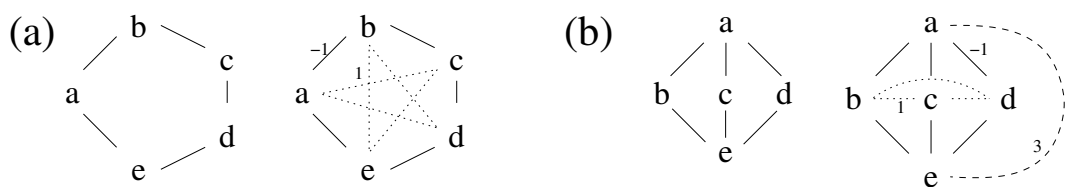


Fig. 2. Diagrams describing the facet-normals of P_5

3 The NJ Cones for Five Taxa

In the case of five taxa there is just one unlabeled tree topology (cf. Fig. 3) and there are 15 distinct labeled trees: We have five choices for the leaf which is not part of a cherry and then three choices how to group the remaining four leaves into two pairs. For each of these labeled topologies, there are two ways in which they might be reconstructed by the NJ algorithm: There are two pairs, any one of which might be chosen in the first step of the NJ algorithm.

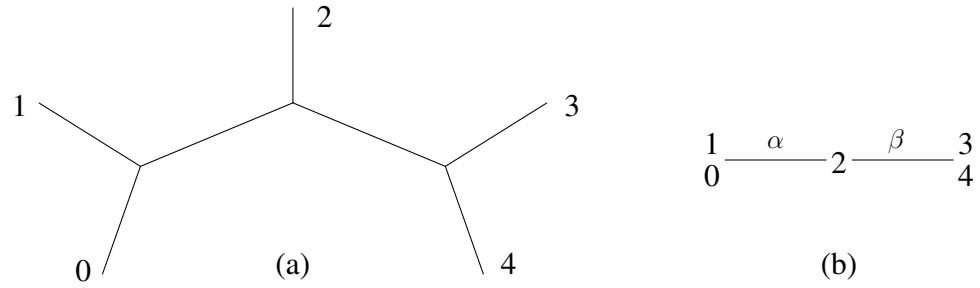


Fig. 3. (a) A tree with five taxa (b) The same tree with all edges adjacent to leaves reduced to length zero. The remaining two edges have lengths α and β.

For distinct leaf labels a, b and $c \in \{0, 1, 2, 3, 4\}$ we define $C_{ab,c}$ to be the set of all input vectors for which the cherry a-b is picked in the first step and c remains as single node not part of a cherry after the second step. For example, the tree in Fig. 3(a) is the result for all vectors in $C_{10,2} \cup C_{43,2}$. Since for each tree topology $((a, b), c, (d, e))$ (this tree topology is written in the Newick format) for distinct taxa $a, b, c, d, e \in \{0, 1, 2, 3, 4\}$, the NJ algorithm returns the same tree topology with any vector in the union of two cones $C_{ab,c} \cup C_{de,c}$, there are 30 such cones in total, and we call the set of these cones $\mathcal{C}$.

3.1 Permuting Leaf Labels

Because there is only one unlabeled tree topology, we can map any labeled topology to any other labeled topology by only changing the labels of the leafs. Such a change of labels also permutes the entries of the distance matrix. In this way, we get an action of the symmetric group S_5 on the input space R^{10}, and the permutation $\sigma \in S_5$ maps the cone $C_{ab,c}$ linearly to the cone $C_{\sigma(a)\sigma(b),\sigma(c)}$. Therefore any property of the cone $C_{ab,c}$ which is preserved by unitary linear transformations must be the same for all cones in $\mathcal{C}$, and it suffices to determine it for just one cone.

The action of S_5 on R^{10} decomposes into irreducible representations by

$$\underbrace{\mathbb{R} \oplus \mathbb{R}^4}_{=S} \oplus \underbrace{\mathbb{R}^5}_{=:W},$$

where the first summand is the subspace of all constant vectors and the second one is the kernel of $A^{(5)}$. The sum of these two subspaces is exactly the space

S generated by the shift vectors. The third summand, which we call W, is the orthogonal complement of S and it is spanned by vectors $w_{ab,cd}$ in W with

$$(w_{ab,cd})_{xy} := \begin{cases} 1 & \text{if } xy = ab \text{ or } xy = cd \\ -1 & \text{if } xy = ac \text{ or } xy = bd \\ 0 & \text{else} \end{cases}$$

where a, b, c and d are pairwise distinct taxa in $\{0, 1, 2, 3, 4\}$ and $(w_{ab,cd})_{xy}$ is the x–yth coordinate of the vector $w_{ab,cd}$. One linearly independent subset of this is

$$w_1 := w_{01,34}, \quad w_2 := w_{12,40}, \quad w_3 := w_{23,01}, \quad w_4 := w_{34,12}, \quad w_5 := w_{40,23}.$$

Note that the 5-cycle (01234) of leaf labels cyclically permutes these basis vectors, whereas the transposition (01) acts via the matrix

$$T := \frac{1}{2} \begin{pmatrix} 2 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & -1 & -1 \\ 0 & -1 & -1 & 1 & -1 \\ 0 & -1 & 1 & -1 & -1 \\ 0 & -1 & -1 & -1 & 1 \end{pmatrix}.$$

Because a five-cycle and a transposition generate S_5, in principle this gives us complete information about the operation.

3.2 The Cone $C_{43,2}$

Since we can apply a permutation $\sigma \in S_5$, without loss of generality, we suppose that the first cherry to be picked is cherry 9, which is the cherry with leaves 3 and 4. This is true for all input vectors $\mathbf{d}$ which satisfy

$$(\mathbf{h}_{9,i}, \mathbf{d}) \geq 0 \text{ for } i = 0, \ldots, 8,$$

where the vector

$$\mathbf{h}_{ij}^{(n)} := -A^{(n)}(\mathbf{e}_i - \mathbf{e}_j)$$

is perpendicular to the hyperplane of input vector for which cherries i and j have the same Q-criterion, pointing into the direction of vectors for which the Q-criterion of cherry i is lower.

We let $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ be the first three rows of $-A^{(4)}R^{(5)}$. If $(\mathbf{r}_1, \mathbf{d})$ is maximal then the second cherry to be picked is 0-1, leaving 2 as the non-cherry node, and similarly $\mathbf{r}_2$ and $\mathbf{r}_3$ lead to non-cherry nodes 1 and 0. This allows us to define the set of all input vectors $\mathbf{d}$ for which the first picked cherry is 3-4 and the second one is 0-1:

$$C_{34,2} := \{\mathbf{d} \,|\, (\mathbf{h}_{9,i}, \mathbf{d}) \geq 0 \text{ for } i = 0, \ldots, 8, \text{ and } (\mathbf{r}_1 - \mathbf{r}_2, \mathbf{d}) \geq 0, (\mathbf{r}_1 - \mathbf{r}_3, \mathbf{d}) \geq 0\}. \tag{7}$$

We have defined this set by 11 bounding hyperplanes. However, in fact, the resulting cone has only nine facets. A computation using `polymake` [6] reveals

that the two hyperplanes $\mathbf{h}_{9,1}$ and $\mathbf{h}_{9,2}$ are no longer faces of the cone, while the other nine hyperplanes in (7) give exactly the facets of the cone. That means that, while we can find arbitrarily close input vectors $\mathbf{d}$ and $\mathbf{d}'$ such that with an input $\mathbf{d}$ the NJ algorithm will first pick cherry 3-4 and with an input $\mathbf{d}'$ the NJ algorithm will first pick cherry 1-2 (or 0-2), we cannot do this in such a way that $\mathbf{d}$ will result in the labeled tree topology of Fig. 3, where 2 is the lonely leaf.

Also note that $\mathcal{C}$, the set of NJ cones which the NJ algorithm returns the same tree topology with any vector in the union of two cones $C_{ab,c} \cup C_{de,c}$, is not convex which is shown in [3]. For details of geometry and combinatorics of the NJ cones for $n = 5$, see [3].

4 The Six Taxa Case

Note that since each of the NJ cones includes constraints for five taxa, the union of the NJ cones which gives the same tree topology is not convex. To analyze the behavior of NJ on distance matrices for six taxa, we use the action of the symmetric group as much as possible. However, in this case we get three different classes of cones which cannot be mapped onto each other by this action. We assume the cherry which is picked in the first step to consist of the nodes 4 and 5. Picking this cherry replaces these two nodes by a newly created node 45, and we have to distinguish two different cases in the second step (see Fig. 4):

- If the cherry in the second step does not contain the new node 45, we may assume the cherry to be 01. For the third step, we again get two possibilities:
  - The two nodes 45 and 01 get joined in the third step. We call the cone of input vectors for which this happens C_{I}.
  - The node 45 is joined to one of the nodes 2 and 3, without loss of generality, to 3. We call the resulting cone C_{II}.
- If the cherry in the second step contains the new node 45, we may assume the other node of this cherry to be 3, creating a new node $45 - 3$. In the third step, all that matters is which of the three nodes 0, 1 and 2 is joined to the node $45 - 3$, and we may, without loss of generality, assume this to be node 2. This gives the third type of cone, C_{III}.

The resulting tree topology for the cone C_{I} is shown in Fig. 5(a), while both C_{II} and C_{III} give the topology shown in 5(b). We now determine which elements of S_6 leave these cones fixed (stabilizer) and how many copies of each cone give the same labeled tree topology:

	C_{I}	C_{II}	C_{III}
stabilizer	$\langle (01), (23), (45) \rangle$	$\langle (01), (45) \rangle$	$\langle (01), (45) \rangle$
size of stabilizer	8	4	4
number of cones	90	180	180
cones giving same labeled topology	6	2	2
solid angle (approx.)	$2.888 \cdot 10^{-3}$	$1.848 \cdot 10^{-3}$	$2.266 \cdot 10^{-3}$

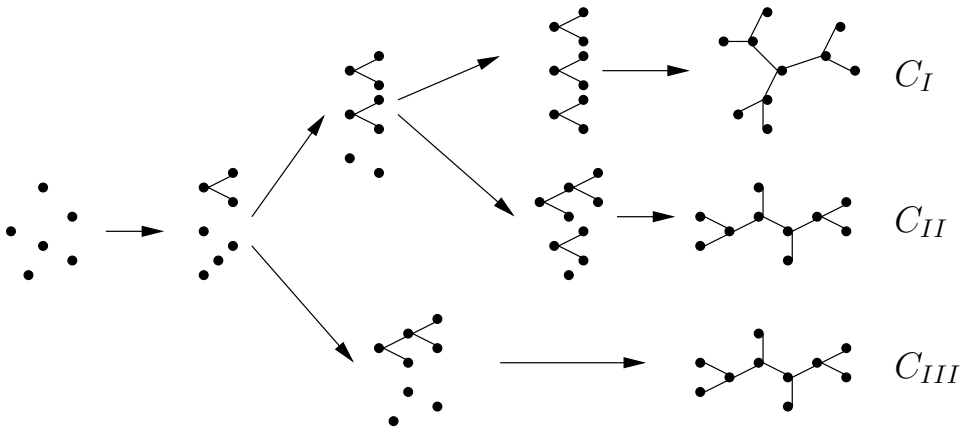


Fig. 4. The three ways of picking cherries in the six taxa case

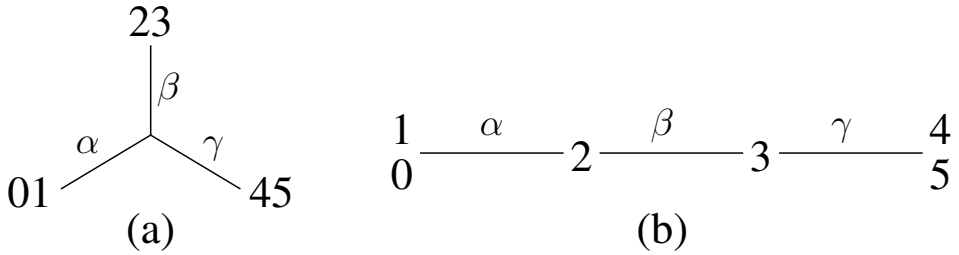


Fig. 5. The two possible topologies for trees with six leaves, with edges connecting to leaves shrunk to zero

Thus, the input space $\mathbb{R}^{15}$ is divided into 450 cones, 90 of type I and 180 each of types II and III. There are 15 different ways of assigning labels to the tree topology in Fig. 5(a), and for each of these there are six copies of C_{I} whose union describes the set of input vectors resulting in that topology. For the topology in Fig. 5(b) we get 90 ways of assigning labels to the leaves, each corresponding to a union of two copies of C_{II} and two copies of C_{III}.

The above table also gives the solid angles of the three cones. In the five taxa case, any two cones can be mapped onto one another by the action of the symmetric group, which is unitary. Therefore all thirty cones have the same solid angle, which must be 1/30. However, in the six taxa case, we get different solid angles, and we see that about "3/4" of the solid angle at the origin are taken by the cones of types II and III. Thus, on a random vector chosen according to any probability law which is symmetric around the origin (e.g., standard normal distribution), NJ will output the tree topology of Fig. 5(b) with probability about 3/4.

On the other hand, any labeled topology of the type in Fig. 5(a) belongs to six cones of type I, giving a total solid angle of $\approx 1.73 \cdot 10^{-2}$, whereas any labeled topology of the type in Fig. 5(b) belongs to two cones each of type II and III, giving a total solid angle of only $\approx 0.82 \cdot 10^{-2}$, which is half as much. This suggests that reconstructing trees of the latter topology is less robust against noisy input data.

5 Simulation Results

In this section we will analyze how the tree metric for a tree and pairwise distances estimated via the maximum likelihood estimation lie in the polyhedral subdivision of the sample space. In particular, we analyze subtrees of the two parameter family of trees described by [13]. These are trees for which the NJ algorithm has difficulty in resolving the correct topology. In order to understand which cones the data lies in, we simulated 10,000 data sets on each of the two tree shapes, T_1 and T_2 (Fig. 6) at the edge length ratio, a/b = 0.03/0.42 for sequences of length 500BP under the Jukes-Cantor model [9]. We also repeated the runs with the Kimura 2-parameter model [7]. They are the cases (on eight taxa) in [13] that the NJ algorithm had most difficulties in their simulation study (also the same as in [10]). Each set of 5 sequences are generated via `evolver` from `PAML` package [17] under the given model. `evolver` generates a set of sequences given the model and tree topology using the birth-and-death process. For each set of 5 sequences, we compute first pairwise distances via the heuristic MLE method using a software `fastDNAml` [12]. To compute cones, we used `MAPLE` and `polymake`.

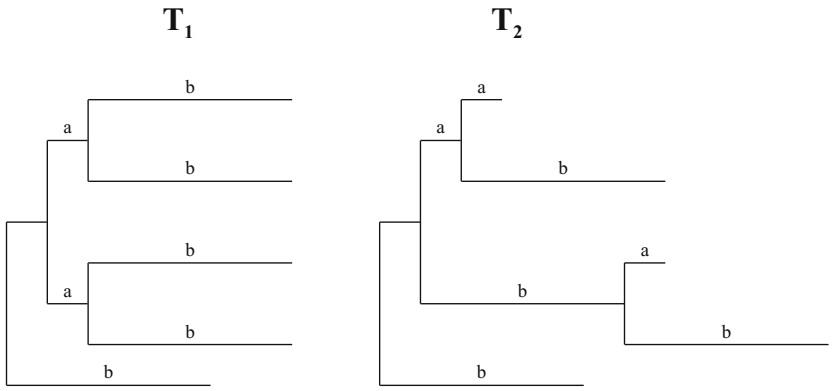


Fig. 6. T_1 and T_2 tree models which are subtrees of the tree models in [13]

To study how far each set of pairwise distances estimated via the maximum likelihood estimation (which is a vector $\mathbf{y}$ in $\mathbb{R}^5$) lies from the cone, where the additive tree metric lies, in the sample space, we calculated the ℓ_2-distance between the cone and a vector $\mathbf{y}$.

Suppose we have a cone C defined by hyperplanes $\mathbf{n}_1, \ldots, \mathbf{n}_r$, i.e.,

$$C = \{\mathbf{x} \mid (\mathbf{n}_i, \mathbf{x}) \geq 0 \text{ for } i = 1, \ldots, r\},$$

and we want to find the closest point in C from some given point $\mathbf{v}$. Because C is convex, for ℓ_2-norm there is only one such point, which we call $\mathbf{u}$. If $\mathbf{v} \in C$ then $\mathbf{u} = \mathbf{v}$ and we are done. If not, there is at least one $\mathbf{n}_i$ with $(\mathbf{n}_i, \mathbf{v}) < 0$, and $\mathbf{u}$ must satisfy $(\mathbf{n}_i, \mathbf{u}) = 0$.

Now the problem reduces to a lower dimensional problem of the same kind: We project $\mathbf{v}$ orthogonally into the hyperplane H defined by $(\mathbf{n}_i, \mathbf{x}) = 0$ and call the new vector $\tilde{\mathbf{v}}$. Also, $C \cap H$ is a facet of C, and in particular a cone, so we proceed by finding the closest point in this cone from $\tilde{\mathbf{v}}$.

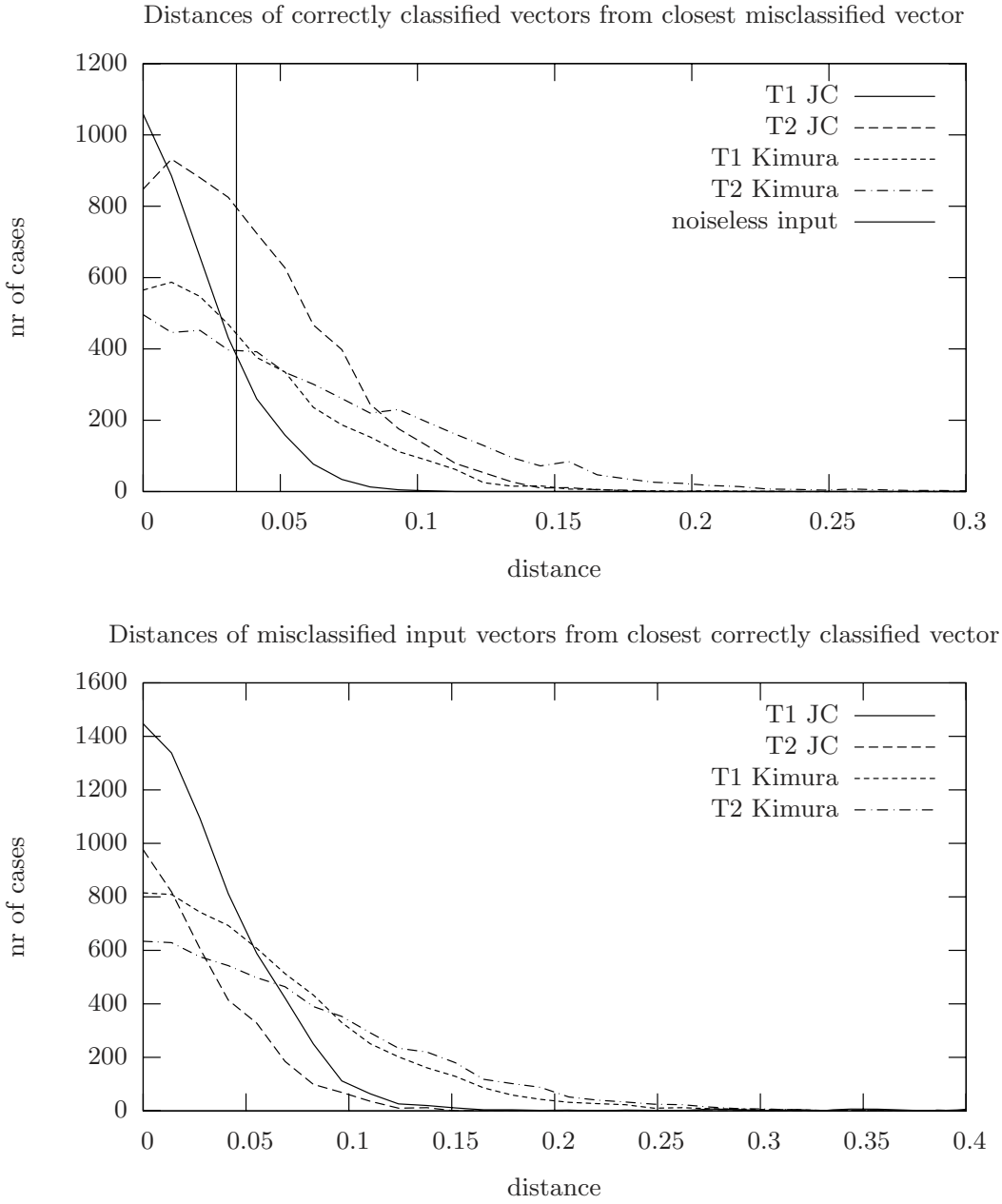


Fig. 7. Distances of correctly (top) and incorrectly (bottom) classified input vectors to the closest incorrectly/correctly classified vector

We say an input vector (distance matrix) is *correctly classified* if the vector is in one of the cones where the vector representation of the tree metric (noiseless input) is. We say an input vector is *incorrectly classified* if the vector is in the complement of the cones where the vector representation of the tree metric is. For input vectors (distance matrices) which are correctly classified by the NJ algorithm, we compute the minimum distance to any cone giving a different tree topology. This distance gives a measure of robustness or confidence in the result, with bigger distances meaning greater reliability. The results are plotted in the left half of Fig. 7 and in Fig. 8. Note that the distance of the noiseless input, i.e., the tree metric from the tree we used for generating the data samples, gives an indication of what order of magnitude to expect with these values.

	JC		Kimura2	
	T1	T2	T1	T2
# of cases	3,581	6,441	3,795	4,467
Mean	0.0221	0.0421	0.0415	0.0629
Variance	$2.996 \cdot 10^{-4}$	$9.032 \cdot 10^{-4}$	$1.034 \cdot 10^{-3}$	$2.471 \cdot 10^{-3}$

Fig. 8. Mean and variance of the distances of correctly classified vectors from the nearest misclassified vector

	JC		Kimura2	
	T1	T2	T1	T2
# of cases	6,419	3,559	6,205	5,533
Mean	0.0594	0.0331	0.0951	0.0761
Variance	0.0203	$7.39 \cdot 10^{-4}$	0.0411	$3.481 \cdot 10^{-3}$

Fig. 9. Mean and variance of the distances of misclassified vectors to the nearest correctly classified vector

For input vectors to which the NJ algorithm returns with a tree topology different from the correct tree topology, we compute the distances to the two cones for which the correct answer is given and take the minimum of the two. The bigger this distance is, the further we are off. The results are shown in the right half of Fig. 7 and in Fig. 9.

From our results in Fig. 8 and Fig. 9, one notices that the NJ algorithm returns the correct tree more often with T_2 than with T_1. These results are consistent with the results in [15,11]. Note that any possible quartet in T_1 has a smaller (or equal) length of its internal edge than in T_2 (see Fig. 6). Gascuel and Steel defined this measure as *neighborliness* [15]. Mihaescu et al. showed that the NJ algorithm returns the correct tree if it works correctly locally for the quartets in the tree [11]. The neighborliness of a quartet is one of the most important factors to reconstruct the quartet correctly, i.e., the shorter it is the more difficult the NJ algorithm returns the correct quartet. Also Fig. 7 shows that most of the input vectors lie around the boundary of cones, including the noiseless input vector (the tree metric). This shows that the tree models T_1 and T_2 are difficult for the NJ algorithms to reconstruct the correct trees. All source code for these simulations described in this paper will be available at authors' websites.

6 Open Problems

Question 1. Can we use the NJ cones for analyzing how the NJ algorithm works if each pairwise distance is assumed to be of the form $D_0 + \epsilon$ where D_0 is the unknown true tree metric, and ϵ is a collection of independent normally distributed random variables? We think this would be very interesting and relevant.

Question 2. With any n, is there an efficient method for computing (or approximating) the distance between a given pairwise distance vector and the boundary

of the NJ optimality region which contains it? This problem is equivalent to projecting a point inside a *polytopal complex* P onto the boundary of P. Note that the size of the complex grows very fast with n. How fast does the number of the complex grow? This would allow assigning a confidence score to the tree topology computed by the NJ algorithm.

References

1. Atteson, K.: The performance of neighbor-joining methods of phylogenetic reconstruction. Algorithmica 25, 251–278 (1999)
2. Bryant, D.: On the uniqueness of the selection criterion in neighbor-joining. J. Classif. 22, 3–15 (2005)
3. Eickmeyer, K., Huggins, P., Pachter, L., Yoshida, R.: On the optimality of the neighbor-joining algorithm. Algorithms in Molecular Biology 3 (2008)
4. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of Molecular Evolution 17, 368–376 (1981)
5. Galtier, N., Gascuel, O., Jean-Marie, A.: Markov models in molecular evolution. In: Nielsen, R. (ed.) Statistical Methods in Molecular Evolution, pp. 3–24 (2005)
6. Gawrilow, E., Joswig, M.: Polymake: a framework for analyzing convex polytopes. In: Kalai, G., Ziegler, G.M. (eds.) Polytopes — Combinatorics and Computation, pp. 43–74 (2000)
7. Kimura, M.: A simple method for estimating evolutionary rates of base substitution through comparative studies of nucleotide sequences. Journal of Molecular Evolution 16, 111–120 (1980)
8. Neyman, J.: Molecular studies of evolution: a source of novel statistical problems. In: Gupta, S., Yackel, J. (eds.) Statistical decision theory and related topics, pp. 1–27. New York Academic Press, London (1971)
9. Jukes, H.T., Cantor, C.: Evolution of protein molecules. In: Munro, H.N. (ed.) Mammalian Protein Metabolism, pp. 21–32. New York Academic Press, London (1969)
10. Levy, D., Yoshida, R., Pachter, L.: Neighbor-joining with phylogenetic diversity estimates. Molecular Biology and Evolution 23, 491–498 (2006)
11. Mihaescu, R., Levy, D., Pachter, L.: Why Neighbor-Joining Works. Algorithmica (2008)
12. Olsen, G.J., Matsuda, H., Hagstrom, R., Overbeek, R.: fastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. Comput. Appl. Biosci. 10, 41–48 (1994)
13. Ota, S., Li, W.H.: NJML: A Hybrid algorithm for the neighbor-joining and maximum likelihood methods. Molecular Biology and Evolution 17(9), 1401–1409 (2000)
14. Saitou, N., Nei, M.: The neighbor joining method: a new method for reconstructing phylogenetic trees. Molecular Biology and Evolution 4, 406–425 (1987)
15. Gascuel, O., Steel, M.: Neighbor-joining revealed. Molecular Biology and Evolution 23, 1997–2000 (2006)
16. Studier, J.A., Keppler, K.J.: A note on the neighbor-joining method of Saitou and Nei. Molecular Biology and Evolution 5, 729–731 (1988)
17. Yang, Z.: PAML: A program package for phylogenetic analysis by maximum likelihood. CABIOS 15, 555–556 (1997)
18. Yang, Z.: Complexity of the simplest phylogenetic estimation problem. Proceedings of the Royal Society B: Biological Sciences 267, 109–116 (2000)
19. Ziegler, G.: Lectures on Polytopes. Springer, Heidelberg (1995)

Neural Algebra and Consciousness: A Theory of Structural Functionality in Neural Nets

Erwin Engeler

Department of Mathematics,
Federal Institute of Technology,
HUT E 31, 8092 Zurich, Switzerland
engeler@math.ethz.ch
http://www.math.ethz.ch/~engeler

Abstract. Thoughts are spatio-temporal patterns of coalitions of firing neurons and their interconnections. Neural algebras represent these patterns as formal algebraic objects, and a suitable composition operation reflects their interaction. Thus, a neural algebra is associated with any neural net. The present paper presents this formalization and develops the basic algebraic tools for formulating and solving the problem of finding the neural correlates of concepts such as reflection, association, coordination, etc. The main application is to the notion of consciousness, whose structural and functional basis is made explicit as the emergence of a set of solutions to a fixpoint equation.

Keywords: Neural nets, combinatory algebra, functional structures, emergent properties, models of consciousness.

1 Introduction

Thoughts are patterns of firing neurons. And so are sensory perceptions, memories, feelings, motor activations, etc. As a population of neurons fires in a pattern it causes the firing of neurons in another pattern, by its neural connections. Thus goes a broadly accepted view of the ongoing activity of the brain. The challenge met by this paper is to find a mathematical framework in which firing patterns M, N, etc. are the basic elements and their composition $M \cdot N = R$ describes how a pattern N which fires in a context M results in a pattern R. Indispensable for such an approach is that the mathematical objects M, etc. have a transparent relation to the basic neurological facts behind a spatio-temporal pattern of firing neurons.

Roughly speaking, each firing pattern is considered as being spread out into parallel tracks of successively firing individual neurons. Each of these tracks is understood as being divided into an initial part and a final part: if the track is part of the spread-out context M and its initial part belongs to N, then its final part belongs to $R = M \cdot N$.

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 96–109, 2008.

The goal of this paper is to cast this rough sketch into a mathematical model and to create the rudiments of a mathematical discipline for it. The proposed neural algebras provide the needed mathematical framework. It will serve for treating the problem of relating the functionality of a neural net to its communication structure in a coherent algebraic fashion.

The examples presented in this paper, in particular our excursion into a theory of consciousness, are necessarily simplistic, but they should show the spirit of such applications: the basic mathematical properties of neural algebras are used to formulate functional-structural problems as equations, and then to solve them algebraically. In particular we can locate neural correlates to some concepts that arise as descriptions of brain functions belonging to a higher level of descriptive language.

2 Neural Algebra

Neural algebras represent sets of neuronal activities – patterns, sequential and spatial, of firing brain cells. These correspond to neural activities as they would show up, for example, in sequences of functional MRI images. The formalization is then used to mirror the composition of such sets – one pattern causing other patterns – as a formal operation on the corresponding elements of the neural algebra.

The fascinating development of neurology, (as impressively told by Eric Kandel, "In Search of Memory" [11]), has allowed mathematical scientists to try approximating the anatomical, physiological and biochemical findings by more or less realistic models, so called artificial neural nets. The early story of these approaches and their relation to artificial intelligence ("connectionism") is collected in Anderson and Rosenfeld [1] reaching up to about 1987. There are other attempts to discuss activities of neural populations mathematically; the most promising so far is probably the dynamical systems approach; others, based on quantum mechanics are somewhat less convincing (see, e.g., the discussion in Koch and Hepp [12]).

Our approach is based on the representation of the activities of neural subpopulations as formal mathematical objects. Deciding on the level of detail about the functioning of neurons and their interconnection determines the interpretation of these expressions. Primarily, a neural net is the directed graph of its synaptic connections. The weakest description level approximates the functionality by providing weights to the synapses and introduces a discrete time behavior of neurons as known from artificial neural nets, familiar for their intuitive appeal, computational richness and developed theory. Our present exposition remains at this level; higher levels of detail might include specific biochemical, and perhaps other mechanisms of communication and distinguish various types of neurons, and might include stochastic elements.

To fix notation, an *artificial neural net* is a weighted directed graph, i.e. a triple (A, L, w), where the set $A = \{a_1, a_2, \dots\}$, which gives the name to the whole net, is a set of elements called *neurons*, connected by directed edges

$L : L \subseteq A \times A$ to which rational weights are attached by the function $w : L \to \mathbf{Q}$, abbreviated $w_{i,j} = w(\langle a_i, a_j \rangle)$. In an active neural net each neuron a_i has at any time instant t an *excitation value* $f(a_i, t) \in \{0, 1\}$, $t \in I$, a subinterval of discrete time $I \subseteq \mathbf{Z}$. These values are interrelated by the structure of the neural net as follows: If $\langle a_1, a_q \rangle, \ldots, \langle a_n, a_q \rangle \in L$, then

$$f(a_q, t+1) = H(\Sigma_i w_{i,q} f(a_i, t)) \,,$$

where H is a $0-1$ valued function: $H(v) = 1$ if $v \geq 1, 0$ otherwise.

Thus, neurons interact by sending information about their excitation states at time t along "axons" via "synapses" to other neurons. The synapses weigh these inputs and the receiving neuron derives from these inputs its excitation state at time $t+1$ according to the function H. At any given moment the weights $w_{i,j}$ attached to the synapse from neuron a_i to neuron a_j are given constants; "learning" may change these values, but this is outside the concern of this paper, although obviously important.

Models somewhat closer to physiological facts than this rather rudimentary one employ real-valued excitation functions and sigmoid functions H, see, e.g., Dehaene et al. [6]. Such models can be incorporated in our approach.

The basic building blocks of our theory are called *track expressions*, denoted by lower case symbols x, y, etc. These formal expressions denote the activation of specific neural connections at specific time instants as follows. For a single neuron a the expression consists of the symbol a alone. If neurons a_1 to a_k have directed edges to neuron a_0 and there is a further edge from a_0 to a_{k+1} and t is any time instant $t \in \mathbf{Z}$ then

$$x := \{a_1, \ldots, a_k\} \xrightarrow[a_0]{t} a_{k+1} \,.$$

is a track expression if the sum of weights of the incoming edges is at least 1.

The set on the left side may be empty, e.g. in the case that a_0 is an "input" neuron with no incoming edges. The neuron a_0 in a sense encodes the activation of this particular connection, it is therefore called the *key neuron* of this expression; formally $a_0 = \nu(x)$. The upper index t of the arrow indicates the time at which the key neuron is activated; formally $t = \tau(x)$. Any one of the a_i, say a_1, may itself be the key neuron of another connection, say

$$y := \{b_1, \ldots, b_s\} \xrightarrow[a_1]{t-1} b_{s+1} \,.$$

Then

$$\left\{ \{b_1, \ldots, b_s\} \xrightarrow[a_1]{t-1} b_{s+1}, \ldots, a_k \right\} \xrightarrow[a_0]{t} a_{k+1}$$

is also a track expression, still with a_0 as its key neuron. More track expressions are obtained by continuing the method of substitution as in the formal definition below.

Let A be any weighted directed graph. We formally define the set of track expressions x together with their key neurons $\nu(x)$ and firing time $\tau(x)$ as a set $S(A)$ of formal elements as follows:

$$
\begin{aligned}
S_0(A) \quad &= A,\ \nu(x) = \ x,\ \ \tau(x) = t \text{ for } x \in A, t \in \mathbf{Z} \\
S_{n+1}(A) = S_n(A) \ \cup \ \{ \ &\{x_1, \ldots, x_k\} \xrightarrow[a_0]{t} x_{k+1} : \textit{if there is an element} \\
&a_0 \in A \textit{ with edges from } a_1, \ldots, a_p \textit{ and to } a_q \textit{, such that} \\
&\tau(x_1), \ldots, \tau(x_k) = t-1, \tau(x_{k+1}) = t+1\,, \\
&\Sigma_i w(\nu(x_i), a_0) \geq 1\,, \\
&\{\nu(x_1), \ldots, \nu(x_k)\} \subseteq \{a_1, \ldots, a_p\} \textit{ and} \\
&\nu(x_{k+1}) = a_q, x_i \in S_n(A),\ i = 1, \ldots, k+1 \,\}\,, \\
\nu(\{x_1, \ldots, x_k\} &\xrightarrow[a_0]{t} x_{k+1}) = a_0\,, \\
\tau(\{x_1, \ldots, x_k\} &\xrightarrow[a_0]{t} x_{k+1}) = t\,.
\end{aligned}
$$

Then $S(A)$, the set of track expressions, is the union of the $S_n(A)$:

$$S(A) = \bigcup_{n=0}^{\infty} S_n(A)\,.$$

Iterated bracketing of track expressions serves to denote neural activities on increasingly higher levels of dependency. As theoretical constructs they are introduced to capture the compositionality of firing patterns and to thus facilitate the construction of an algebraic superstructure on a given neural net, the neural algebras.

We now come to the formal definition of firing patterns. Let A be a neural net, considered as a directed graph (equipped with further data in case of higher level detail as above), and let $S(A)$ be the set of all track expressions. A set M of track expressions constitutes a *firing pattern*, if, loosely speaking, it corresponds to a temporal pattern of firing neurons in A. Formally this means that there is an assignment of an excitation function $f(a_i, t)$ to the set of neurons a_i and firing times t occurring in the track expressions in M such that $f(\nu(x), \tau(x)) = 1$ for all track expressions x in M, and such that this assignment conforms to the firing laws.

Firing patterns are designed to be identified with (mental) functions as their *neural correlates*. This rests on the fact that certain subnets can be understood as having specific functionalities based on them. This singling out of subnets and firing patterns based on them is a virtual, theoretical, superstructure on the neural net and is typically guided by hypotheses on their function such as receiving sensory input or analysing activities based on some other subnet, etc. Research in neurology has resulted in an enormous and growing knowledge base of such facts for humans and for some animals.

Our mathematical framework identifies neural correlates as firing patterns; this makes it possible to capture the *compositionality* of such neural correlates

quite generally, as follows: Let the neural firing pattern M be based on a subgraph A_M of A, the *support* of M. Then M may take account of activities N supported by subgraph A_N and produce activities R, supported by A_R.

The activation of N allows the activation of key neurons in M, which in turn results, due to the structure of M, in the activation of the firing pattern designated by R. Mathematically this situation corresponds to a composition operation $M \cdot N = R$. Formally, we have the following definition:

$$M \cdot N = \{\, x : \textit{there is an element } \{x_1, \ldots, x_k\} \xrightarrow[a]{t} x \textit{ in } M \textit{ such that } \{x_1, \ldots, x_k\} \subseteq N \,\}.$$

Note that whenever M and N are firing patterns, then so is $M \cdot N$. This definition captures the rôles that the neurons in A_M and A_N play: Indeed, the neuron a has an activation history that depends on the histories of $\nu(x_1), \ldots, \nu(x_k)$ and influences that of $\nu(x)$.

It is in the nature of the things, that R itself may again be an "initiation" or an "action", etc. Indeed, each firing pattern can be used as a left multiplier, representing a law of interaction, or as a right multiplier, representing the input to the interaction. In this way, the set of firing patterns associated to a neural net A constitute an algebraic structure, the *Neural Algebra* $\mathcal{N}_A$.

3 Some Mathematical Background

Let A be a neural net. Let $F(A)$ be the set of firing patterns of A; this set is provided with the composition operation defined in the previous section, thus constituting an algebraic structure $\mathcal{N}_A = \langle F(A), \cdot \rangle$. It is in these algebras that we are to solve the equations describing interactions between neural processes formulated as firing patterns. Each one of these may of course contain neural populations that are not used in the composition operation and may, if we imagine them in nature, be physically far removed except for the overlapping necessary for the composition.

There are three useful theorems about neural algebras:

Theorem 1 (Fixpoint Theorem)**.** *In $\mathcal{N}_A$ all* fixpoint equations *have a solution; the solutions form a lattice by inclusion.*

Theorem 2 (Embedding Theorem)**.** *$\mathcal{N}_A$ is a subalgebra of a* combinatory algebra*, indeed, it is a combinatory algebra for certain nets A.*

Theorem 3 (Representation Theorem)**.** *If A is a sufficiently rich directed graph and Φ is a binary relation over B, a subset of A, then Φ is representable in $\mathcal{N}_A$ using an embedding f defined by: a and b are in the relation Φ if and only if $f(a) \cdot f(b) = f(b)$, where f maps B into $S(A)$.*

To prove Theorem 1, first note the monotonicity of the algebraic operation $M \cdot N$:

If $N_1 \supseteq N_2$ then $M \cdot N_1 \supseteq M \cdot N_2$ by the definition of the operation; equally $M_1 \cdot N \supseteq M_2 \cdot N$ for $M_1 \supseteq M_2$. Hence, if $\varphi(X)$ is any algebraic composition of

X with elements of $F(A)$ then $X' \supseteq X$ implies $\varphi(X') \supseteq \varphi(X)$. More generally, if D is a directed set of elements of $\mathcal{N}_A$ then $\varphi(\bigcup D) = \bigcup_{X \in D} \varphi(X)$. From this follows, that the fixpoint equation $\varphi(X) = X$ has a least solution $\bigcup_n \varphi^n(\emptyset)$, where $\varphi^0(X) = X$ and $\varphi^{n+1}(X) = \varphi(\varphi^n(X))$.

Namely: $\emptyset \subseteq \varphi(\emptyset) \subseteq \varphi(\varphi(\emptyset)) \subseteq \ldots$ is a directed set, hence $\varphi(\bigcup_n \varphi^n(\emptyset)) = \bigcup_n \varphi^{n+1}(\emptyset) = \bigcup_n \varphi^n(\emptyset)$; thus $\bigcup_n \varphi^n(\emptyset)$ is a fixpoint of $\varphi(X)$. In fact, it is the smallest fixpoint.

The above solution method for fixpoint equations can be expanded to simultaneous equations, e.g. in the case of the *coordination problem* (finding inputs that coordinate two activities M and N):

$$M \cdot P = Q\,, \qquad\qquad N \cdot Q = P\,.$$

Given M and N as known, the simultaneous fixpoint equations can be solved in $\mathcal{N}_A$ by a generalization of the method for one fixpoint: Let $P_1 = N \cdot \emptyset$, $Q_1 = M \cdot \emptyset$, $P_{n+1} = N \cdot Q_n$, $Q_{n+1} = M \cdot P_n$. Then $P = \bigcup_i P_i$, $Q = \bigcup_i Q_i$ are (least) solutions.

To simplify exposition, we occasionally drop the firing-time superscripts. We also may use the same track variables at different ocurrencies in an expression, the superscripts are thought of being supplied conforming to the actual positions.

To prove Theorem 2, let A be a complete directed graph: each node is connected to all nodes and all edges are weighted 1. Then the neural algebra $\mathcal{N}_A$ is a combinatory algebra, which means that it has the property that for any algebraic expression $\varphi(X_1, \ldots, X_k)$ in variables $X_1, \ldots, X_k$ there exists an element T in $\mathcal{N}_A$ for which $(\ldots((TM_1)M_2)\ldots M_k) = \varphi(M_1, \ldots, M_k)$ for all values $M_1, \ldots, M_k$ of $X_1, \ldots, X_k$. The object T is defined by

$$T := \{(\alpha_1 \to_{a_1} (\alpha_2 \to_{a_2} \ldots (\alpha_k \to_{a_k} x))) : x \in \varphi(\alpha_1, \ldots, \alpha_k)\,,$$

$$\alpha_1, \ldots, \alpha_k \subseteq S(A)\,,\ \textit{finite}\}\,.$$

It is traditionally called the *combinator* associated to the algebraic expression φ. To verify the case $k = 2$, consider $(TX_1)X_2 = \varphi(X_1, X_2)$:

$$\begin{aligned}(TM_1)M_2 &= \{x : \exists\alpha \subseteq M_1\,, \exists\beta \subseteq M_2\,, (\alpha \to_a (\beta \to_b x)) \in T\} \\ &= \{x : \exists\alpha \subseteq M_1\,, \exists\beta \subseteq M_2\,, x \in \varphi(\alpha, \beta)\}\,.\end{aligned}$$

The last equation follows from

$$\varphi(M_1, M_2) \;=\; \bigcup\{\varphi(\alpha, \beta) : \alpha \subseteq M_1\,,\ \beta \subseteq M_2\,,\ \alpha, \beta \text{ finite}\}\,.$$

proving Theorem 2.

The proof of Theorem 3 consists of verifying the following set-recursive definition of the mapping f, for all a and b in the relation Φ:

$$f(a) = \{a\} \cup \{\{b\} \longrightarrow_a x : b \in B, x \in f(a)\}\,.$$

Neural algebras, as we have defined them, are related to combinatory algebras, as shown above. The latter have evolved from beginnings in mathematical logic,

namely Lambda Calculus and the related Combinatory Logic [2,4]. While these subjects were created in the 1930's with the foundations of mathematics as their aim, they have had considerable influence in theoretical computer science, especially after Scott and Plotkin constructed their well-known models of the Lambda Calculus. (For a concise introduction see Engeler [10], Chapter 3.) The basis of the present work is a richer type of models, the subject of a prolonged effort of the author and his students at the ETH: "The Combinatory Programme", [9]. This research program deals with a large variety of mathematical subjects, including universal algebra and computer algebra, and later set theory and category theory.

Neural algebras as they arise from natural examples do not have complete graphs, although they have very large numbers of synaptic connections indeed. In applications, therefore, the neurons needed for realizing firing patterns like the T above in the proof of Theorem 2 may have to be obtained by enlarging the underlying graph A to B; "recruiting new neurons and synapses" as we may say. Mathematically speaking, this corresponds to an algebraic extension $\mathcal{N}_B$ of the original $\mathcal{N}_A$ which then contains the new element. This is a construction familiar in algebra, where to solve equations it may be necessary to expand the algebraic structure, e.g. from rational to algebraic numbers. In natural neural nets, such expansions may conceivably consist in mobilizing already present but partially dormant neurons and connections.

The case of *concordance* of activities is an example: if U and V are two firing patterns, their intersection $U \cap V$ describes their functional and temporal concordance, the extent to which U and V concur. $U \cap V$ is in fact the result of applying the operator $\wedge$ on them:

$$\wedge UV = (\wedge \cdot U) \cdot V = U \cap V\,,$$

with

$$\wedge = \{\{x\} \longrightarrow_r \{\{x\} \longrightarrow_s x\} : x \in S(B)\}\,,$$

where r and s are two newly recruited neurons. The union of two firing patterns can be obtained in a similar manner.

The solution of equations other than fixpoint equations is a challenging mathematical problem. Indeed, it can be shown that all degrees of computational complexity and of unsolvability can occur. Some solution methods based on algebraic extensions of a combinatory algebra have been described in Chapter III of [9], but much work needs to be done and experience gathered from applications.

4 In Search of Consciousness

One possible application of our theory is in the search for neural correlates of mental functions. Let us turn to the entirely speculative case of "consciousness", with the goal of analyzing the well-known thesis that consciousness is the power of self-reflection. The definition of consciousness as self-reflection is just one of a long and involved history of attempts to define this concept. In this, we are well aware of the caveat of Francis Crick: "Until the problem [of consciousness]

is understood much better, any attempt at a formal definition is likely to be either misleading or overly restrictive, or both," [3]. For one thing, the sheer size and complexity of that network precludes any complete analysis. More fundamentally: There can be no effective method to decide whether a neural net, once initiated, will develop a given response. (This can easily be shown by simulating Turing Machines in neural nets).

However, it is quite possible to work out attributions of content and function to types of neural structures; in the case of consciousness this results in the *Structure Theorem* below.

Let us then understand neural consciousness as *the ability of a neural net B ("the brain") to consciously observe itself as being conscious and as consciously planning and acting.* These abilities are embodied as activities in sub-populations of the "brain", to be represented here by firing patterns; their interrelation is expressed by their composition: If C is the firing pattern corresponding to "consciousness", and M_1, M_2, etc. are the firing patterns corresponding to the context of observing, acting, planning, moving, etc. then $M_1 \cdot C$, $M_2 \cdot C$, etc. are the results of observing, acting, etc. as dependent on consciousness. To the sum of these results, together with C itself, C is again applied. Translated into neural algebra, our definition of *consciousness* transforms into an equation of the form

$$C \cdot \left(C \cup \bigcup_i M_i \cdot C \right) = C \,.$$

The solutions of this fixpoint equation constitute the set of persistent activity patterns in a net of neurons that may be understood as states of "consciousness". (The apparent circularity of our non-formal definition thus resolves itself as multiple entry of the unknown in a single equation).

Obviously, the *quality of consciousness* in this formal sense depends crucially on the size and structure of the underlying "brain" and on the degree of involvement of other brain functions, such as memory, language, intentionality. This is reflected in the fact that different such activity patterns are (lattice-)ordered by inclusion and correspond to to different forms or to emerging stages of consciousness. Thus, forms of consciousness may already be found in neural nets of primitive animals, and indeed even in neural simulations of computers.

Basically, our results say that consciousness is always based on one or more recurrent loops of active neurons and feeds forward from these to other activated regions of the brain; patterns that are solely based on stimulus-and-response cannot support consciousness. The various forms of consciousness depend in this way on the richness and the activity of the mind as embodied in the neural net constituting the brain.

4.1 The Structural Basis of Consciousness

To obtain the structural basis of consciousness, we solve the fixpoint equation *structurally*, which means that we "disregard time". Formally, this means that

$$\{x \underset{a_1}{\longrightarrow} y : x \in A_3,\, y \in A_2\}$$

is to be read as

$$\{x \xrightarrow[a_1]{t} y : x \in A_3,\, y \in A_2,\, t \in \mathbf{Z}\}\,.$$

Computing consciousness fixpoints shows up the structural facts that are relevant in all models of the brain. We observe the following facts:

Theorem 4 (Structure Theorem of Consciousness)
(a) Consciousness always has a base in one or more cycles of the directed graph.
(b) Consciousness can be expanded along any outgoing edge.
(c) Consciousness never expands backwards into cycle free "stimulus and response" subgraphs.

To prove (a) consider the simple case of a cycle of neurons a_1, a_2, a_3, cyclically connected with weights 1. Assume this cycle embedded in a graph with an edge of weight 1 leading from a neuron b to a_1, and one from a_3 to a neuron c, again with weight 1.

Let

$$A_1 = \{a_1\} \cup \{x \underset{a_1}{\longrightarrow} y : x \in A_3, y \in A_2\}\,,$$
$$A_2 = \{a_2\} \cup \{x \underset{a_2}{\longrightarrow} y : x \in A_1, y \in A_3\}\,,$$
$$A_3 = \{a_3\} \cup \{x \underset{a_3}{\longrightarrow} y : x \in A_2, y \in A_1\}\,,$$
$$C = A_1 \cup A_2 \cup A_3\,,$$
$$B = C \cup \{b \underset{a_1}{\longrightarrow} y : y \in A_2\} \cup \{x \underset{a_3}{\longrightarrow} c : x \in A_2\}\,.$$

Then

$$C \cdot C = A_1 \cdot C \cup A_2 \cdot C \cup A_3 \cdot C = C\,,$$

and

$$B \cdot C = C \cdot C \cup \emptyset \cup \{c\}\,.$$

Hence

$$C \cdot (C \cup B \cdot C) = C \cdot (C \cup \{c\}) = C \cdot C = C\,.$$

Items (b) and (c) can be as easily proved: (b) is exemplified by extending C to $C' = C \cup \{x \underset{a_3}{\longrightarrow} c : x \in A_2\}$; (c) by observing that $C = C \cup \{b \underset{a_1}{\longrightarrow} y : y \in A_2\}$ is not a fixpoint.

4.2 The Emergence of Consciousness

Consciousness "simply happens" in any sufficiently rich neural net. It is a typical example of an *emerging phenomenon*. Mathematically, emergence consists in the approximation of a fixpoint: Assume that we already have a fixpoint C (the empty set $\emptyset$ is always available), and let E extend C and N extend M. Using E and N we can gradually progress to a new fixpoint C' as follows:

Define $\varphi_Y(X) = X \cdot (X \cup Y \cdot X)$ and observe
$\varphi_M(C) = C \subseteq E \subseteq \varphi_N(E) \subseteq \varphi_N(\varphi_N(E)) \subseteq \ldots$ is a directed set, hence $\varphi_N(\bigcup_n \varphi_N^n(E)) = \bigcup_n \varphi_N^{n+1}(E) = \bigcup_n \varphi_N^n(E)$; thus $C' = \bigcup_n \varphi_N^n(E)$ is a fixpoint of $\varphi_N(X)$.

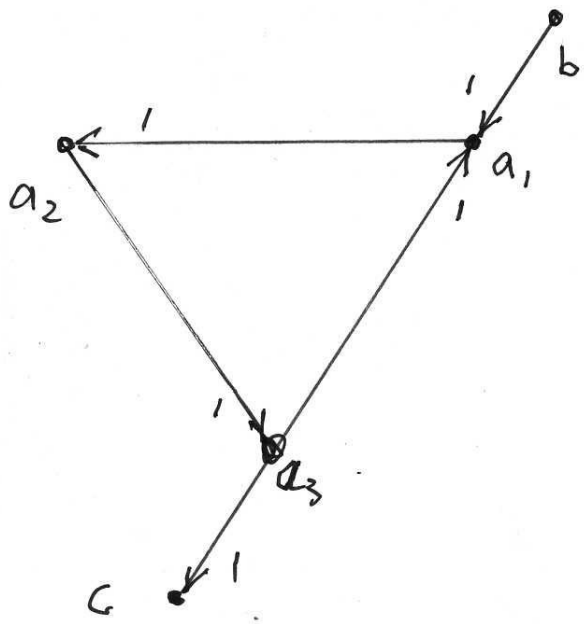


Fig. 1. A cycle of consciousness

For a somewhat plausible example, imagine a chess player whose sensory system provides him with the positions in the game, upon which his planning faculty decides on a plan, e.g. a particular endgame. The action coordination determines the first move, which the motorics of the player transforms into moving a particular piece. There follow updates of the planner about that move and to remember to wait for the challenge for his next move.

Using this context as a guide, consider a *microbrain* $\mathcal{N}_B$, the neural algebra over the directed graph B, representing the neural substrate which supports firing patterns which we name, to fix ideas, I for sensory input, V for vision, P for perception and planning, A for activation of actions, M for motor activity, L for language, S for speech, H for body perception. In our microbrain, these firing patterns have as key neurons just one neuron each, namely i, v, p, a, m, l, s, h, connected and weighted according to the diagram in Figure 2. The structural laws for the firing patterns of $\mathcal{N}_B$ can be read off the diagram in Figure 2 as follows in the form of simultaneous recurrence equations:

$$H = \{h\} \cup \{x \longrightarrow_h y, x \longrightarrow_h z : x, y \in H, z \in A\}$$
$$A = \{a\} \cup \{x \longrightarrow_a y, x \longrightarrow_a z, u \longrightarrow_a z, \{x, u\} \longrightarrow_a z_1, \{x, u\} \longrightarrow_a z_2$$
$$: u \in H, x, y \in P, z_1 \in M, z_2 \in L\}$$
$$M = \{m\} \cup \{x \longrightarrow_m y : x \in A, y \in V\}$$
$$V = \{v\} \cup \{x \longrightarrow_v y, z \longrightarrow_v y, \{x, z\} \longrightarrow_v y : x \in M, y \in P, z \in I\}$$
$$P = \{p\} \cup \{x \longrightarrow_p y, z \longrightarrow_p y, z \longrightarrow_p u, x \longrightarrow_p u : x, y \in A, z \in V, u \in L\}$$
$$L = \{l\} \cup \{x \longrightarrow_l z, y \longrightarrow_l z, \{x, y\} \longrightarrow_l z : x \in P, y \in A, z \in S\}$$
$$I = \{i\}, S = \{s\}$$

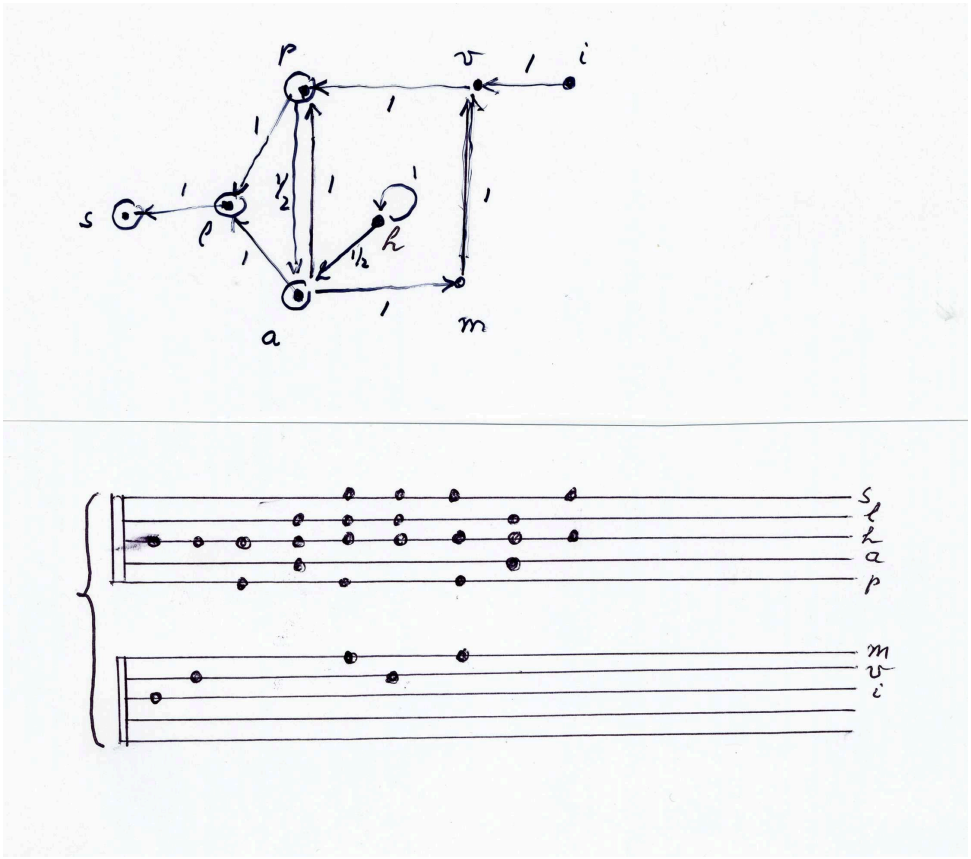


Fig. 2. The microbrain $\mathcal{N}_B$ and its scores

The consciousness equation will admit at least one but conceivably many solutions in $\mathcal{N}_B$. Expanding them, as suggested by *emergence*, may add further ones.

The minimal solution is, of course, the empty set $\emptyset$, since $\emptyset \cdot X$ is the empty set for all X.

A first nontrivial solution is $C_0 := H_0$ "the brain may only be conscious of the beating heart". Let, again recursively,

$$H_0 = \{h\} \cup \{h : x \longrightarrow_h y : x, y \in H_0\}\,.$$

To verify that H_0 supported by the cyclic subgraph on the neuron h is indeed a fixpoint follows at once from (a). Other fixpoints follow as consequences of (b) and (c).

As observed above, consciousness-fixpoints in $\mathcal{N}_B$ form a lattice under the inclusion relation and are as a rule not all ordered in one sequence. They are best described by the supporting subgraphs of these patterns, e.g.

$$C_1 = A_1 \cup P_1 \cup H_1\,,$$

based on key neurons h, a and p, where

$$A_1 = \{a\} \cup \{\{x, y\} \longrightarrow_a z : y, z \in P_1, x \in H_1\}\,,$$
$$P_1 = \{p\} \cup \{x \longrightarrow_p y : x, y \in A_1\}\,,$$
$$H_1 = \{h\} \cup \{x \longrightarrow_h y, x \longrightarrow_h z : x, y \in H, z \in A_1\}\,.$$

Remaining with microbrains, we may also incorporate other ideas about the functioning of the brain. For example, we may interpolate a neuron w between p and v, which makes the system "watch out" for particular input. Or we may establish an edge between v and m for immediate reactive movements, etc.

Even in rudimentary neural algebras such as $\mathcal{N}_B$, there develops a rich variety of conscious behavior. The activation history of such a net may be looked at as a sort of musical score for the "theme" that the brain plays, see Figure 2 for an example. The activation of neurons on which consciousness is based is shown at the top of the score, the bottom would be something like (relative) subconscious.

Visualization of consciousness as "a sort of orchestral piece played in the brain" points out a connection with the findings of the school of Wolf Singer (e.g. [8]), which shows the central importance of synchronicity. It also shows the importance of the persistence of activation patterns for the constitution of a sustained consciousness, the "self" in a "*sin*-phonic" view, possibly with a recognizable personal "style".

5 Discussion

To belabor the obvious: as a model of the human brain, the microbrain $\mathcal{N}_B$ is unrealistic by about twelve orders of magnitude; the neuron v for example stands for something like the visual cortex, and m may be a cascade of interrelated neural nets, etc.

However, our findings about the existence of the lattice of consciousness activity complexes scale up to "brains" of all levels of complexity, and we may speculate, whether "core consciousness" and levels of "extended consciousness", as described in the literature (e.g. by [5]), correspond to such fixpoints. Also, we may speculate about the history of activations of these different forms of consciousness and whether this might involve moving up from body-awareness such as C_1 based on lower fixpoints to higher fixpoints. Experiences by introspection might just be based on such migrations, which, by the way, may have to move through higher or lower points in the lattice to reach from one to the other. Other such experiences (e.g. so-called "earworms" of popular melodies), as well as observations on periodicities of brain activities in observed conscious behaviors, seem to match the musical-score paradigm of brain activity.

In neural algebra, thoughts, emotions, communication, etc., are just elements to be computed with, this is all there is, *formally*. Remaining with the musical score paradigm, the notes of a Mozart piece would analogously be all there is! But, far from formality being an impoverishment of these concepts, the mathematical approach presents an unending challenge, of which we now sketch only a few immediate aspects.

5.1 On Laws of Thought

When Boole created an algebraic discipline for computing with truth values of statements, "thoughts" were understood as being expressed in language, and

parts of the grammar of language provided the patterns of the algebraic operations. Should we not now take firing patterns as "thoughts" and their composition as the algebraic operations? This results, abstracting from the unknown complexities of human neural algebras, in a position which we could call *neural logic*, regarding neural algebra as the true algebra of thoughts.

If we aim to understand mental activities as compositions of firing patterns, there are several basic concepts that need neural correlates. Further research would have to show, whether, for example, the following proposals have a chance:

(a) To classify a firing pattern X as conforming to a template F, as in recognizing a face, we could simply compose the two and consider the result as indicating in what, and how far, the classification holds true. Or, taking the basic idea from the Representation Theorem 3, we might identify true classification with $F \cdot X = X$.

(b) In Theorem 2 it is shown that complex composition patterns of objects can be considered as objects in a combinatory algebra. In the present context, we might consider the notion of analogy as a particular firing pattern, say L. Then L expresses the fact that the notion X is to the leading example U as it is to the analogon V, thus:

$$LXUV = \wedge \cdot (X \cdot U) \cdot (X \cdot V)\,,$$

using the intersection operator from Section 3.

(c) Combinatory algebra has objects that correspond to natural numbers and to computing with them; our model of the brain inherits this. Although this shows that a rich enough $\mathcal{N}_B$ can handle all computable functions, and indeed simulate any Turing machine, it is implausible that the published versions of arithmetic in combinatory algebra (as in [10]) are the ones realized in the human brain, (cf., for example, Dehaene [7] on numerosity). More generally, "intelligence" may be closely related to the easy availability of combinatory objects, templates, which represent basic forms of relatedness, such as analogy, causality, duality, etc.

5.2 Extended and Collective Consciousness

In Section 4 we have taken the simplest case of composition of "consciousness" with the structure of the mind, namely

$$C \cdot \left(C \cup \bigcup_i M_i \cdot C \right) = C\,.$$

The "mind" $M = \bigcup_i M_i$ may allow artificial extensions $M' \supset M$, which may conceivably induce an extension of consciousness. The use of tools "till they become a part of ourselves" is a striking example. In designing tools, e.g. software tools, it may therefore be of interest to keep in mind the neural connections that we have identified with the structural functioning of consciousness.

There is no reason to restrict the present approach to neural nets that are "brains". Equally well we could consider populations of agents, for example a colony of ants, that operate under certain constraints and with certain well-defined schemes of communication. Thus would emerge concepts of collective consciousness for such populations; and it would be attractive to try and develop this idea in a variety of contexts. In a similar vain, the human brain itself is in fact a population of such individual agents, neurons, whose collaboration may have evolved by a learning process, including the recruitment of new members and new connections.

Acknowledgments. I wish to express my thanks to Klaus Hepp, who challenged me on my bold statements that I knew "the right mathematics for modelling the brain and consciousness." In consequence, the present approach was worked out and presented in a preliminary form at the "Brain Fair Zurich" in 2005. Of course, all responsibilities for this work remain mine.

References

1. Anderson, J.A., Rosenfeld, E.: Neurocomputing. MIT Press, Cambridge (1988)
2. Church, A.: A Set of Postulates for the Foundations of Logic. Annals of Math. 33, 346–366 (1932)
3. Crick, F.: The Astonishing Hypothesis. Simon & Schuster, Ltd (1994)
4. Curry, H.B.: Grundlagen der Kombinatorischen Logik. Amer. J. of Math. 52, 509–536, 789–834 (1929)
5. Parvizi, J., Damasio, A.: Consciousness and the Brainstem. Cognition 79, 135–159 (2001)
6. Dehaene, St., Sergent, C., Changeux, J.-P.: A Neural Network Model Linking Subjective Reports and Objective Physiological Date During Conscious Perception. Proc. Nat. Acad. Sci. 100, 8520–8525 (2003)
7. Dehaene, St., Molko, N., Cohen, L., Wilson, A.J.: Arithmetic and the Brain. Current Opinion in Neurobiology 14, 218–224 (2004)
8. Engel, A.K., Fries, P., Singer, W.: Dynamic Predictions and Synchronicity. Nature Reviews Neuroscience 2, 704–716 (2001)
9. Engeler, E., Aberer, K., Gloor, O., von Mohrenschildt, M., Otth, D., Schwärzler, T., Weibel, T.: The Combinatory Programme. Birkhäuser, Boston (1995), `http://www.math.ethz.ch/~engeler`
10. Engeler, E.: Foundations of Mathematics, ch.3. Springer, New York (1983) also in Russian, Chinese and German
11. Kandel, E.R.: In Search of Memory, Norton New York (2006)
12. Koch, C., Hepp, K.: Quantum Mechanics in the Brain. Nature 40, 1011–1012 (2006)

An Algorithm for Qualitative Simulation of Gene Regulatory Networks with Steep Sigmoidal Response Functions

Liliana Ironi[1], Luigi Panzeri[1], and Erik Plahte[2]

[1] IMATI - CNR, via Ferrata 1, 27100 Pavia, Italy
[2] CIGENE (Centre for Integrative Genetics) and Department of Mathematical Sciences and Technology, Norwegian University of Life Sciences, P.O. Box 5003, N-1432 As, Norway

Abstract. A specific class of ODEs has been shown to be adequate to describe the essential features of the complex dynamics of Gene-Regulatory Networks (GRN). But, the effective exploitation of such models to predict the dynamics of specific GRNs by classical numerical schemes is greatly hampered by the current lack of precise and quantitative information on regulation mechanisms and kinetic parameters. Due to the size and complexity of large GRNs, classical qualitative analysis could be very hard, or even impracticable, to be carried out by hand, and conventional qualitative simulation approaches rapidly lead to an exponential growth of the generated behavior tree that, besides all possible sound behaviors, may also contain spurious ones. This paper discusses the work-in-progress of a research effort aiming at the design and implementation of a computational framework for qualitative simulation of the dynamics of a class of ODE models of GRNs. The algorithm we propose results from a set of symbolic computation algorithms that carry out the integration of qualitative reasoning techniques with singular perturbation analysis methods. The former techniques allow us to cope with uncertain and incomplete knowledge whereas the latter ones lay the mathematical groundwork for a sound and complete algorithm capable to deal with regulation processes that occur at different time scales.

Keywords: Gene regulatory network, qualitative simulation, singular perturbation analysis.

1 Introduction

A variety of modeling formalisms, ranging from directed graphs over Boolean networks to differential models, ordinary and partial differential equations, along with related simulation algorithms have been applied to study gene regulatory systems as demonstrated by a rich literature, and discussed in several monographs and survey papers. The mathematical aspects of such approaches, and the evaluation of their relative strengths and weaknesses are discussed in [1].

A growing number of theoretical as well as experimental papers show that gene regulation is threshold-dependent, i.e. only effective above or below a certain threshold. ODE models with switch-like interaction terms allow us to provide detailed descriptions of gene regulatory mechanisms at the molecular level [2], and, in theory, they

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 110–124, 2008.

could be used to make numerical predictions of the network behavior and its response to environmental stimuli. But, in practice, even when the system at hand is very well studied, their exploitation meets several obstacles: (i) models are often large, complex and above all, nonlinear, and traditional pen and paper analysis could be inadequate or too time-consuming; (ii) numerical simulations are seriously hampered by the current lack of precise and quantitative information on the biochemical reaction mechanisms underlying regulatory interactions, kinetic parameters and threshold concentrations.

The qualitative analysis and simulation of the dynamics of GRNs is a rather appropriate solution as, in the current state of knowledge, the key issue is to understand how specific activity patterns derive from given network structures, and what different types of dynamical behaviors are possible. However, due to the complexity of GRN models, to carry out analytically a complete qualitative study of their dynamics might be very hard, or even impracticable. Nor is the recourse to conventional qualitative simulation approaches [3], developed within the Artificial Intelligence research framework to cope with the need to represent and predict the dynamics of systems characterized by incomplete knowledge, an appropriate solution. Such approaches can deal with generic classes of dynamical systems, and generate, starting from an initial system state, the whole range of system dynamics. Each qualitative behavior is generated by applying transition rules that are grounded on mathematical tools actually too simple to compensate for the lack of complete knowledge. This results in a number of drawbacks, e.g. their inability to upscalability, the exponential growth of the generated behaviors, and the generation of spurious behaviors, that reveal to be particularly serious in predicting nonlinear dynamics of regulatory networks even in the case of networks with a small number of interacting genes. Thus, the need for the development of *ad hoc* computational frameworks for qualitative analysis and simulation of GRN models.

A first effort in this direction is given by a qualitative simulator, called GNA [4]. It is based on the integration of Qualitative Reasoning (QR) concepts [3] and control theory methods to cope with both incomplete knowledge and threshold-dependent regulation mechanisms. GNA assumes that threshold-regulated response functions are step functions, discontinuous in the threshold hyperplanes. On the one hand, such an assumption considerably simplifies the analysis as the model results in piecewise-linear equations. On the other hand, it raises the problem to find a proper continuous solution across the threshold hyperplanes, or, in other word, to seek for generalized solutions of ODEs with discontinuous right-side terms. Among the possible definitions of generalized solutions, GNA adopts the Filippov one [5], that is quite popular and convenient in a control context, but may fail when applied to approximate the limit solutions of a continuous model. This together with a further approximation introduced in the GNA algorithm for computational purposes might not always guarantee its soundness and completeness [6].

These problems are avoided if the response functions are sigmoidal and vary continuously from zero to one with a steep rise around the threshold. The ensuing dynamics is both linear and nonlinear with different time scales, but has been analysed using singular perturbation theory [7]. Here we present a qualitative simulation algorithm for models with steep sigmoid response functions based on this work. Despite an incomplete knowledge of parameter values, we construct all possible trajectories and their associated parameter space domains, starting from an initial state and parameter space

domain, by using QR key concepts suitably revised, and by iteratively exploiting symbolic computation procedures.

2 A Modeling Framework for the Study of GRN Dynamics

Recent experimental findings as well as theoretical justifications (see [8] for references) seem to support the following generic and phenomenological dynamic model for a GRN:

$$\dot{x_i} = f_i(\mathbf{Z}) - \gamma_i x_i, \tag{1}$$

where the dot denotes time derivative, x_i is the concentration of gene product number i, $i = 1, \ldots, n$, γ_i is the relative decay rate of x_i, $\mathbf{Z}$ is a vector with Z_{jk} as components, and $Z_{jk} = S(x_j, \theta_{jk}, q)$ is a sigmoid or binary (i.e. Heaviside or step) response function with threshold θ_{jk} and steepness parameter q. The thresholds associated with each x_i are ordered according to $\theta_{ij} < \theta_{ik}$ if $j < k$. Finally, $\mathbf{x}(t, \mathbf{x^0}, q)$ is the solution satisfying the initial condition $\mathbf{x}(0, \mathbf{x}^0, q) = \mathbf{x}^0$.

The differentiable functions f_i are regulatory production functions, frequently composed by algebraic equivalents of Boolean functions [9]. Eq. (1) is assumed to catch the essential features of a wide range of regulatory systems, where the regulatory control may be at the level of transcription, mRNA stability, translation, or post-translation. The state variables may be concentrations of proteins, hormones, mRNA, and intracellular ions [10]. The framework has been applied to model real world networks: the initiation of sporulation in *B. subtilis* [11], and the response to nutritional stress and carbon starvation in *E. coli* [12,13], using Heaviside response functions. We assume the sigmoid functions are very steep Hill functions, viz. $S(x, \theta, q) = x^{1/q}/(x^{1/q} + \theta^{1/q})$, where $0 < q \ll 1$. Under a number of reasonable assumptions a solution $\mathbf{x}(t, \mathbf{x^0}, q)$ of Eq. (1) starting in an arbitrary initial point $\mathbf{x}^0$ at $t = 0$ can be uniformly approximated by the zero order solution $\mathbf{x}(t, \mathbf{x^0}, 0)$. The derivation is based on singular perturbation theory.

Eq. (1) could be applied to GRNs of any size and complexity, and the generic analysis developed in [7,14] is fully applicable to networks of any size. However, as the network becomes large, there is a combinatorial explosion of phase-space domains and parameter combinations that need to be investigated, and a computerized, algorithmic approach becomes a necessity. That is the first motivation for the present work. Our second motivation is that sigmoidal response functions in many cases are more realistic than step functions, and could be preferred for modeling real systems.

Regular and Singular Stationary Points. The threshold hyperplanes $x_i = \theta_{ij}$ divide phase-space into regular and switching domains. A *regular domain* D_R (also called a *box*) is an open rectangular domain between threshold planes in which the values of all sigmoids are close to 0 or 1. In a box we put $Z_{ij} = B_{ij}$. In a *switching domain* D_S the x_i are divided into two disjoint sets: the switching variables x_s and the regular variables x_r, $s \in \mathcal{S}$, $r \in \mathcal{R}$, where $\mathcal{S} \cup \mathcal{R} = \mathcal{N} = \{1, 2, \ldots, n\}$. A x_s is very close to one of its thresholds, while a x_r lies in the open domain between two adjacent thresholds. Thus, a switching domain is a narrow boundary layer surrounding a section of a threshold plane or an intersection of threshold planes. The union of all the regular domains is denoted Δ_R, while Δ_S is the union of all the switching domains, and $\Delta = \Delta_R \cup \Delta_S$.

A stationary point $\mathbf{P}(q)$ is called a *regular stationary point* (RSP) if it is located in a box, and a *singular stationary point* (SSP) if it is located in a switching domain. The zero order approximation $\mathbf{P}^0$ of a RSP $\mathbf{P}(q)$ is a solution of $f_i(\mathbf{B}) - \gamma_i x_i = 0$, $i \in \mathcal{N}$, where $\mathbf{B}$ represent the Boolean values of the Z-variables. This is trivial to solve for each $\mathbf{B}$. A RSP is always asymptotically stable.

If a SSP exists in a D_S, it is in lowest order found as the solution $\mathbf{P}^* = (x_r^*, \theta_s)$ of

$$\begin{aligned} f_r(B_r, Z_s) - \gamma_r x_r &= 0, \quad r \in \mathcal{R}, \\ f_s(B_r, Z_s) - \gamma_s \theta_s &= 0, \quad s \in \mathcal{S}. \end{aligned} \tag{2}$$

In the following we make the apparently realistic and simplifying

Assumption $\mathcal{A}$. Every x_i only regulates one gene at each of its thresholds.

Because each Z_{ij} then only occurs in a single rate function, the second of Eqs. (2), which would otherwise represent a set of polynomial equations that could be very hard to solve, is reduced to a set of linear equations. Then, there is at most a single stationary point $\mathbf{P}^*$ in each D_S. A necessary condition for a solution of the second equation is that there is precisely one Z-term in each equation and that these terms produce a non-singular Jacobian matrix $\mathbf{J}_s = \partial f_s / \partial Z_s$. These Jacobian elements form a loop $\mathcal{L}$ with loop product L. A suitable renumbering leads to a block-diagonal $\mathbf{J}_s$, where each block is a permutation matrix associated with a sub-loop $\mathcal{L}_j$ of $\mathcal{L}$. Then the characteristic equation $|\mathbf{J}_s - \lambda I| = 0$ is

$$\prod_{j=1}^{m} \Big((-\lambda)^{l(j)} + L_j \Big) = 0 \tag{3}$$

where m is the number of sub-loops of $\mathbf{J}_s$, $l(j)$ is the length of $\mathcal{L}_j$ and L_j is the loop product of $\mathcal{L}_j$. It follows that $\mathbf{P}^*$ has no eigenvalues with positive real part and is stable iff (i) $l(j) = 1$ or $l(j) = 2$ for all j, (ii) $L_j < 0$ for $j = 1$, (iii) $L_j > 0$ for $j = 2$, the latter case occuring if there is a negative loop among the two variables, giving a pair of imaginary eigenvalues. All other cases will give an eigenvalue with a positive real part.

Thus, when the SSP is stable, we only encounter all eigenvalues with negative real part in the very special case when $\mathbf{J}_s$ is diagonal with only negative diagonal elements. If some $l(j) = 2$, we get a pair of purely imaginary eigenvalues. This causes a problem, because the standard proof of the validity of the singular perturbation approximation requires eigenvalues with negative real parts. This remains to be investigated. However, numerical simulations indicate that singular perturbation should work also in this case.

Dynamical Behavior. In a box all x_i are regular, and in the step function limit the solutions approach the solutions of

$$\dot{x_r} = f_r(\mathbf{B}) - \gamma_r x_r. \tag{4}$$

In a switching domain D_S where $x_s = \theta_{s,s_j}$, the rapid motion of the switching variables is described by

$$Z'_{s,s_j} = D_{s,s_j}[f_s(B_r, Z_{s,s_j}) - \gamma_s \theta_{s,s_j}], \tag{5}$$

where $D_{s,s_j} = Z_{s,s_j}(1 - Z_{s,s_j})/\theta_{s,s_j}$ stems from the derivative of the Hill function, and the prime denotes differentiation with respect to $\tau = t/q$.

If Eq. (5) has no stationary and asymptotically stable solution $Z^*_{s,s_j} \in (0,1)^\sigma$, where $\sigma = |\mathcal{S}|$, the system just passes through D_S and leaves $(0,1)^\sigma$ in an *exit point* at the boundary, with no change in the regular variables. Otherwise, the Z-variables come to rest in the exit point Z^*_{s,s_j}, and the slow motion of the regular variables x_r in this D_S is described by the linear and independent equations

$$\begin{aligned} \dot{x_r} &= f_r(B_r, Z^*_{s,s_j}) - \gamma_r x_r \ , \\ Z_{s,s_j} &= Z^*_{s,s_j} \ , \end{aligned} \tag{6}$$

until a stable state is reached or the solution leaves D_S and passes into an adjacent switching domain. In the singular perturbation language, Eq. (5) is called the *boundary layer equation* and Eq. (6) the *reduced equation*.

The phase space of Eq. (5) is confined to the so-called Z-cube $\mathcal{Z}(D_S) = [0,1]^\sigma$ associated with D_S. In the limit $q \to 0$, the interior of the Z-cube describes the motion of the singular Z-variables, while the faces describe the switching variables in the adjacent switching domains, and the vertices represent the adjacent boxes [14]. Each Z-cube has a set of *entrance* and *exit* points where trajectories can enter, respectively exit from, the interior of the Z-cube. An entrance point is also an exit point of an adjacent domain, so we will only need exit points. An exit point can be located on the boundary of the cube or be an internal point.

According to singular perturbation theory [15], the solutions of Eqs. (5,6) taken together approximate the exact solution in D_S for q close to zero, i.e. for steep sigmoids [7]. One just has to express x_s by the solution $Z_{s,s_j}(\tau)$ and replace τ by t/q. For each switching domain D_S there is essentially only one problem: to determine the exit point Z^*_s where the trajectory leaves D_S or the system comes to rest. The details of the motion of the Z-variables are not important as they only occur in a narrow part of phase space and in a negligible time span. Using this approach, we can construct a solution starting in an arbitrary initial point $\mathbf{x}^0$ at $t = 0$ through any finite sequence of regular and switching domains by supplying Eqs. (4-6), solved for each domain encountered, with initial conditions which ensure a continuous trajectory. This zero order solution $\mathbf{x}(t, \mathbf{x^0}, 0)$ is then a uniform approximation to $\mathbf{x}(t, \mathbf{x^0}, q)$ for $0 < q \ll 1$. This is the theoretical basis for the algorithm to be described below.

3 Qualitative Simulation of GRN Models

Our work aims at the development of a qualitative simulation algorithm based on sophisticated mathematical tools, and specifically tailored to capture network dynamical properties that are invariant for ranges of values of kinetic parameters. To this end, in the following we revise and *ad hoc* tailor the key concepts underlying qualitative simulation algorithms to our specific class of models.

Qualitative Value. The partition of the whole system domain, induced by the ordered sets, Θ_i, of the n_i symbolic threshold values θ_{ij} associated with each x_i, identifies qualitatively distinct *n*-dimensional hyper-rectangles D that define *the system qualitative values*. Let us observe that, to characterize switching domains, instead of the sharp value θ_{ij}, we consider a range of values around it, whose width, $\delta > 0$, is a monotonic

function of the steepness parameter q with $\delta(q) \to 0$ for $q \to 0$. Let us denote by $\underline{\theta}_{ij}$ and $\overline{\theta}_{ij}$ the values $\theta_{ij} - \delta/2$ and $\theta_{ij} + \delta/2$, respectively. Then, each D results from the product of intervals, either all open, $(\overline{\theta}_{ij}, \underline{\theta}_{i(j+1)})$, or at least one closed, $[\underline{\theta}_{i(j+1)}, \overline{\theta}_{i(j+1)}]$.

Qualitative State. Let $A(D)$ be the set of domains adjacent to $D \in \Delta$. The *qualitative state* of D, $QS(D) = \{D_k \mid D_k \in A(D),\ D \to D_k\}$, is defined by all of its adjacent domains D_k towards which a transition from it is possible. Each transition from D identifies a domain next traversed by a system trajectory. More precisely, if we number by i the domain D traversed at time t_i, each of its successors $D_k \in QS(D)$ will be traversed by different trajectories at time t_{i+1}.

State Transitions. The possible transitions from D are determined by different strategies according to whether $D \in \Delta_R$ or $D \in \Delta_S$. In the former case, like in traditional QR methods and in GNA [4], transitions are determined by the signs of $\dot{x}_i$. As $\dot{x}_i$ are defined by linear expressions, such signs are easily determined by exploiting the inequalities that define the parameter space domain, and constrain the RSPs to belong to specific domains. In the case, $D \in \Delta_S$, a sign-based strategy is not practicable as the expressions for $\dot{x}_i$ are nonlinear. A convenient way to proceed is given by singular perturbation analysis: transitions from D towards adjacent D_k are determined by the locations of its exit points that can be either on (i) the boundary of D or in (ii) its interior. Except in the case (ii), the number of exit points may be greater than one. Then, in general, the successors of D are not uniquely determined. But, through symbolic computation procedures, it is possible to calculate the set of inequalities, I_j^i, on parameters that hold when a transition from D_i to D_j occurs. Then, each path from D_i to D_j is clearly identified by the 3-tuple $\langle D_i, D_j, I_j^i \rangle$.

Qualitative Behavior. A finite sequence of paths, where each path is clearly both linked and consistent with its predecessor and successor, defines a *qualitative behavior*:

$QB = \langle D_0, I_0 \rangle, \langle D_0, D_1, I_1^0 \rangle, \ldots, \langle D_k, D_i, I_i^k \rangle, \ldots, \langle D_F, I_F \rangle$, where D_0 is the initial domain, and D_F either contains a stable fixed point or identifies a cycle, i.e it is an already visited domain. I_0 is the initial set of inequalities that defines the parameter space domain, and I_F the set of inequalities on parameter values associated with D_F.

3.1 The Simulation Algorithm

Given as input, (i) n symbolic state equations of the form (1); (ii) n ordered sets $\Theta_i = \{\theta_{ij}\}$; (iii) an initial domain $D_0 \in \Delta$; (iv) a set of symbolic inequalities I_0 on parameter values defining a parameter space domain PSD_0, the simulation algorithm generates all possible state transitions, and represents them by a directed tree rooted in D_0, $\mathrm{BT}(D_0)$. In such a tree the vertices correspond to D_i, and the arcs, labeled by the inequalities I_j^i, to the transitions from D_i to D_j. Each branch in $\mathrm{BT}(D_0)$ defines a qualitative trajectory from D_0, that occurs when the values of parameters satisfy its related inequalities. Such a trajectory is characterized by the traverse of specific domains, and abstracts all those numeric solutions of the ODE, obtained with different values of either the initial condition $\mathbf{x}^0 \in D_0$ or parameters, that cross the same domains. The main steps of the algorithm, sketched in Fig. 1, are summarized below:

1. *Partition* the phase space into regular and switching domains.
2. *Calculate* the qualitative state $QS(D_i)$ of the current domain D_i.
3. *Determine constraints* I_k^i on parameters for each path $e_{ik} = D_i \rightarrow D_k$, where $D_k \in QS(D_i)$.
4. *Append* $\langle D_i, D_k, I_k^i \rangle$ to BT(D_0) if I_k^i are consistent with the initial constraints I_0, and *mark* D_i as visited domain.
5. *Repeat* from step 2 for each D_k.

In the following we detail step 2, that is the core of the algorithm.

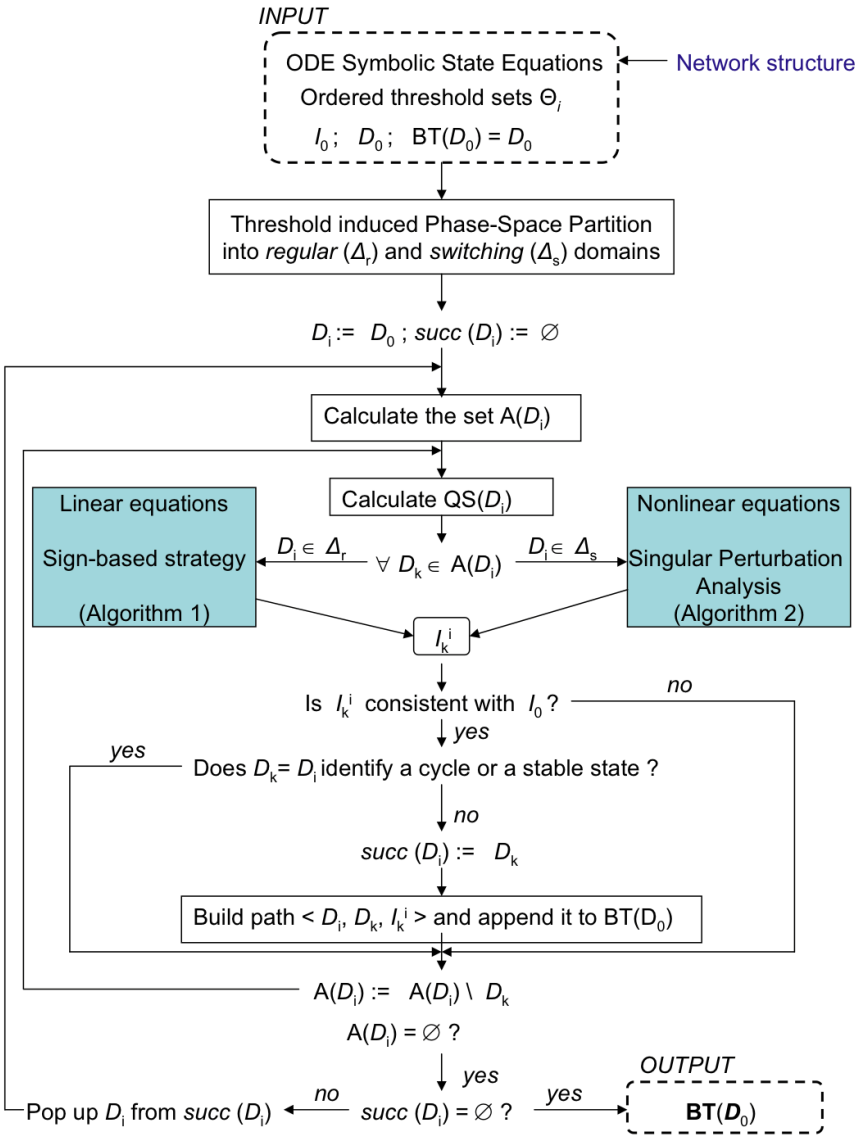


Fig. 1. Main steps of the simulation algorithm

3.2 Calculation of the Qualitative State

The calculation of the qualitative state requires two separate algorithms to implement the different strategies adopted according to whether $D_i \in \Delta_R$ or $D_i \in \Delta_S$. Both algorithms calculate the conditions on parameters I_k^i that are consistent with I_0,

$\forall D_k \in QS(D_i)$. Let us define I_k^i *consistent* with I_0 when it defines a not empty parameter space domain PSD_k^i such that $PSD_k^i \subseteq PSD_0$. Furthermore, we define the relative position of D_1 with respect to D_2, indicated by $V(D_1, D_2) = \{v_j\}_{j=1}^n$ where $v_j \in \{-1, 0, 1\}$, by the comparison of the intervals defining D_1 and D_2.

Transition from a Regular Domain (Algorithm 1). The algorithm that constructs the possible paths from regular domains is, in principle, similar to that one proposed by GNA, but it is more informative as it calculates the I_k^is. From now on, for the sake of simplicity, we indicate the two consecutive thresholds $\theta_{ji}, \theta_{j(i+1)}$ by θ_j, θ'_j. Then, let D_i be defined by $D_i = \prod_{j=1}^n (\overline{\theta}_j, \underline{\theta'_j})$. In outline, the algorithm performs the following steps:

1. *Calculate $A(D_i)$ and state equations in D_i.* The algorithm calculates the set $A(D_i)$, the symbolic state equations in D_i, and the focal point $\mathbf{x}^*$ towards which all trajectories head when $t \to \infty$.
2. *Calculate I_k^i and possible transitions.* $\forall D_k \in A(D_i)$, the algorithm calculates the set of inequalities on parameters I_k^i that need to be fulfilled to have a transition from D_i to D_k. As all the equations are linear in D_i, such inequalities are calculated by imposing that the signs of state variable rates match the relative position of D_k with D_i. Let $V(D_k, D_i) = \{v_j\}_{j=1}^n$ be the relative position of D_k with respect to D_i. I_k^i is given, $\forall j \in \{1, \ldots, n\}$, by either the inequality $x_j^* > \overline{\theta}'_j$ if $v_j = 1$ or $x_j^* < \underline{\theta}_j$ if $v_j = -1$. Thus, if the calculated inequality set defines a not empty parameter space domain $PSD_k^i \subseteq PSD_0$, then a transition towards D_k is possible and the qualitative state $QS(D_i)$ is updated accordingly.
3. *Check the existence of a* RSP *in D_i.* A stable point RSP exists in D_i, i.e. $D_i \in QS(D_i)$, if $\widetilde{PSD} \subseteq PSD_0$ and $\widetilde{PSD} \neq \emptyset$, where $\widetilde{PSD}$ is a parameter space domain defined by the set of inequalities $\overline{\theta}_j < x_j^* < \underline{\theta'_j}, \forall j \in \{1, \ldots, n\}$.

Algorithm 1. Calculate $QS(D_i)$ for Regular Domain

1: Set $I_k^i \leftarrow I_0$
2: **for all** $D_k \in A(D_i)$ **do**
3: *Calculate* $V(D_k, D_i) = \{v_j\}_{j=1}^n$
4: **for** $j = 1$ to n **do**
5: *Calculate* the symbolic state equation in D_i and its stationary solution x_j^*;
6: *Update*:

$$I_k^i \leftarrow I_k^i \wedge \begin{cases} (x_j^* > \overline{\theta}'_j) & \text{if } v_j = 1 \\ (x_j^* < \underline{\theta}_j) & \text{if } v_j = -1 \end{cases}$$

7: **if** $PSD_k^i \neq \emptyset$ **then**
8: *Append* D_k to $QS(D_i)$ and *label* the path $D_i \to D_k$ by I_k^i.
9: *Append* D_i to $QS(D_i)$ if $\widetilde{PSD} \subseteq PSD_0$ and $\widetilde{PSD} \neq \emptyset$, where $\widetilde{PSD}$ is a parameter space domain defined by the set of inequalities $\{(\overline{\theta}_j < x_j^* < \underline{\theta}'_j)\ \forall j \in \{1, \ldots, n\}$

Transition from a Switching Domain (Algorithm 2). The nonlinear dynamics in a switching domain D_i is characterized by fast and slow motions, respectively associated with x_s and x_r that are independently calculated.

Algorithm 2. Calculate $QS(D_i)$ for a Switching Domain

Initialize $EP \leftarrow \emptyset$
Calculate symbolically the boundary layer system in $\mathcal{Z}(D_i)$
Update EP by adding all the vertices of $\mathcal{Z}(D_i)$
Calculate symbolically the Jacobian matrix $\mathbf{J}$
for all $F \in \mathcal{F}$ **do**
  Calculate $\mathbf{J_F}$
  if there is a complete loop in $\mathbf{J_F}$ **then**
    Calculate the stationary point in F by solving symbolically $\mathbf{Z'_s} = 0$
    Update EP by adding the point calculated at the previous step
for all $\tilde{\mathbf{Z}}^{\mathbf{k}} = \{\tilde{Z}_s^k\} \in EP$ **do**
  Initialize $I_k^i \leftarrow I_0$
  for $s = 1$ to $\sigma(D_i)$ **do**
    if $\tilde{\mathbf{Z}}^{\mathbf{k}} \in F, \quad F \in \mathcal{F}$ **then** *Build* $I_k^i \leftarrow I_k^i \wedge (0 < \tilde{Z}_s^k < 1)$
  for $j = 1$ to m **do**
    if $lenght(l_j) > 2$ **then** *Remove* $\tilde{\mathbf{Z}}^{\mathbf{k}}$ from EP
    if $length(l_j) = 1$ **then**
      if $L_j < 0$ **then** *Build* $I_k^i \leftarrow I_k^i \wedge (L_j < 0)$ **else** *Remove* $\tilde{\mathbf{Z}}^{\mathbf{k}}$ from EP
    else
      if $length(l_j) = 2$ **then**
        if $L_j > 0$ **then** *Build* $I_k^i \leftarrow I_k^i \wedge (L_j > 0)$ **else** *Remove* $\tilde{\mathbf{Z}}^{\mathbf{k}}$ from EP
  for all $l \in \mathcal{L}_F = \{l : l \in \{1, \ldots, \sigma(D_i)\}, \quad \tilde{Z}_l^k \in \{0, 1\}\}$ **do**
    Build

$$I_k^i \leftarrow I_k^i \wedge \begin{cases} (Z'_l(\tilde{\mathbf{Z}}^{\mathbf{k}}) > 0) & \text{if } \tilde{Z}_l^k = 1 \\ (Z'_l(\tilde{\mathbf{Z}}^{\mathbf{k}}) < 0) & \text{if } \tilde{Z}_l^k = 0 \end{cases}$$

  Calculate the Exit Domain Set: $ED = \{D_k : D_k = \Sigma_{D_i}^{-1}(\tilde{\mathbf{Z}}^{\mathbf{k}}), \tilde{\mathbf{Z}}^{\mathbf{k}} \in EP\}$
  Calculate symbolically in $\tilde{\mathbf{Z}}^{\mathbf{k}}$ the reduced system and its stationary solution $\mathbf{x}^*$
  for all $D_k \in ED$ **do**
    Calculate $V(D_k, D_i) = \{v_j\}_{j=1}^n$
    for $r = \sigma(D_i) + 1$ to n **do**
      Build

$$I_k^i \leftarrow I_k^i \wedge \begin{cases} (x_r^* > \theta'_r) & \text{if } v_r = 1 \\ (x_r^* < \theta_r) & \text{if } v_r = -1 \end{cases}$$

    if $PSD_k^i \neq \emptyset$ **then** *Append* D_k to $QS(D_i)$ and *label* the path $D_i \rightarrow D_k$ with I_k^i
Append D_i to $QS(D_i)$ if $\exists EP \in int(\mathcal{Z}(D_i))$ and $\widetilde{PSD} \subseteq PSD_0$ and $\widetilde{PSD} \neq \emptyset$ where $\widetilde{PSD}$ is defined by the set of inequalities $\overline{\theta}_j < x_j^* < \underline{\theta}'_j \; \forall j \in \{\sigma(D_i) + 1, \ldots, n\}$

The study of the fast dynamics is performed in $\mathcal{Z}(D_i)$ in the scaled time τ, and aims at localizing the set of exit points in $\mathcal{Z}(D_i)$ rather than at detailing the dynamics within it. Such points clearly identify the next domains the trajectories are moving towards from D_i along the x_s directions. To this end, the algorithm proceeds as follows:

1. *Calculate the boundary layer equations in* D_i. The algorithm symbolically calculates the boundary layer equations in the Z variables, and defines the mapping

$\Sigma_{D_i} : \mathcal{D}_i \rightarrow \mathcal{Z}(D_i)$, where $\mathcal{D}_i = D_i \cup A(D_i)$, that states a correspondence between D_i and its adjacent domains D_k with the interior and the elements on the boundary of $\mathcal{Z}(D_i)$. Let $\mathcal{F}$ be the set of both the faces and the interior of $\mathcal{Z}(D_i)$: its generic element $F = \Sigma_{D_i}(D)$, $D \in \Delta_s$ is either a face of $\mathcal{Z}(D_i)$ when $D \in A(D_i)$ or its interior when $D = D_i$.

2. *Search for stationary points.* Let us denote by EP the set of stationary points, initially made up of the vertices of $\mathcal{Z}(D_i)$. The set of the candidate exit points EP is updated by the possible stationary point on each element of $\mathcal{F}$. To this end, the algorithm symbolically calculates, $\forall F \in \mathcal{F}$, the Jacobian matrix $\mathbf{J}_F$. As the presence of a non-zero loop is a necessary condition for the existence of a stationary point, the algorithm first searches for a non-zero loop involving all variables in $\mathbf{J}_F$: in case, it symbolically calculates the stationary point on F, and updates accordingly the set of candidate exit points EP.
3. *Calculate I_k^i and possible transitions by checking stability of stationary points.* The inequality set I_k^i is calculated for each candidate exit point $\tilde{\mathbf{Z}}^{\mathbf{k}} = \{\tilde{Z}_s^k\} \in EP$ by requiring that each point fulfills stability conditions. In addition, for those $\tilde{\mathbf{Z}}^{\mathbf{k}}$ located on elements of $\mathcal{F}$, I_k^i is further constrained by the inequalities on parameters that impose $0 < \tilde{Z}_s^k < 1$ for each $\tilde{Z}_s^k \notin \{0, 1\}$. The stable points located on $\mathcal{Z}(D_i)$ clearly identify the set of all possible exit domains, i.e. those domains towards which a transition from D_i is possible. Such domains are easily calculated by applying the map Σ^{-1} to each element of $\mathcal{Z}(D_i)$ that contains an exit point.

The slow dynamics of regular variables x_r is studied in the normal time in the usual frame of reference, and it is reconstructed from the reduced system through the same symbolic procedure given for regular domains.

3.3 Remarks about Symbolic Computations

Most of the calculations are performed symbolically. In addition to plain symbolic algebraic manipulation like arithmetic and derivative operations, the algorithms are required to tackle more complex tasks, such as: (i) refine an inequality set with an another one; (ii) check the consistency of two sets of inequalities I_1 and I_2; (iii) solve systems of equations; (iv) find loops in the Jacobian matrix.

Assumption $\mathcal{A}$ leads to major simplifications. As for (iii), Eq. (1) are generally multilinear in $\mathbf{Z}$, but now assume a linear form in each D_S, and can be straightforwardly solved and analyzed for stability. Also, the solution of problems (i) and (ii) are simplified as the inequalities are always linear. Thanks to algorithms proposed both by the literature and common symbolic computation package, such as Mathematica [16], the tasks (i)–(iii) are simplified and feasible. As for the task (iv), it is performed by using cycle–detection algorithms and tools of matrix graph theory [17].

In a more general modeling framework where Assumption $\mathcal{A}$ is removed, it could be really very hard to solve symbolically both the inequalities and equations as they might result in polynomials with very high order even for low dimensional systems.

4 An Example of the Algorithm at Work

To illustrate the algorithm at work, let us consider as an example the ODE system:

$$\begin{aligned} \dot{x_1} &= \kappa_1(1 - Z_{11})(1 - Z_{22}) + \kappa_2(1 - Z_{21}) - \gamma_1 x_1 \ , \\ \dot{x_2} &= \kappa_3(1 - Z_{12}) - \gamma_2 x_2 \ , \end{aligned} \tag{7}$$

where the Z_{jk} are expressed by Hill functions, parameters are all strictly positive and the quantity spaces $\Theta_1 = \{0 < \theta_{11} < \theta_{12} < \overline{x}_1\}$, $\Theta_2 = \{0 < \theta_{21} < \theta_{22} < \overline{x}_2\}$ partition the phase space into domains as showed in Fig. 3(a).

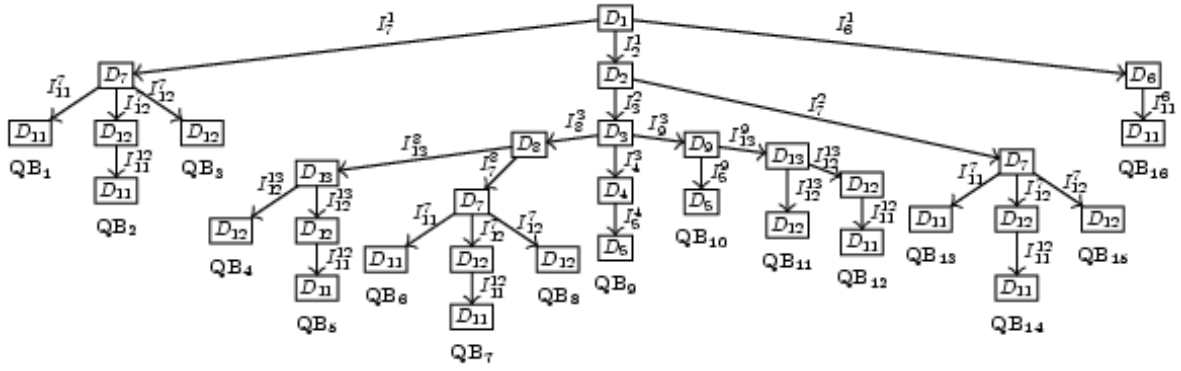


Fig. 2. Behavior tree rooted in D_1

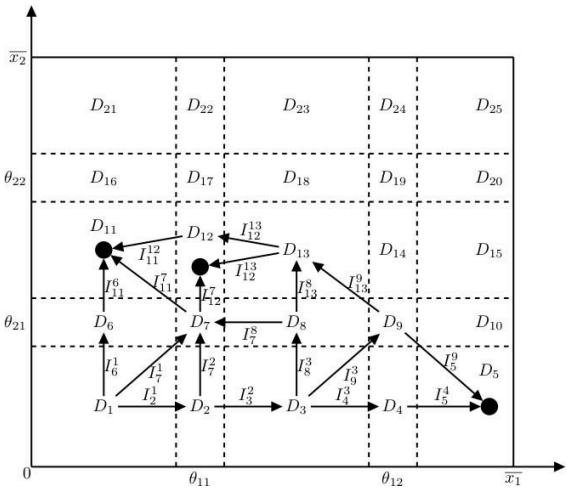
The simulation starts from D_1 with I_0 defined as follows:

$$I_0 : (\frac{\kappa_1 + \kappa_2}{\gamma_1} > \overline{\theta}_{11}) \quad \wedge \quad (\overline{\theta}_{21} < \frac{\kappa_3}{\gamma_2} < \underline{\theta}_{22}) \ . \tag{8}$$

The algorithm builds the behavior tree showed in Fig. 2, and calculates the inequalities on parameters, listed in Fig. 3(b), that are associated with each path in BT(D_1). Three reachable stable states, located in D_{11}, D_{12} and D_5, are identified by the final leaf of each branch in BT. As $D_{12} \in \Delta_s$, one of them is a SSP whereas the others are RSPs. These stable states are reached by different predicted qualitative behaviors, each of them occurring under specific constraints on parameters. For example, the trajectory QB_{16} starting from D_1, crossing D_6, and reaching a RSP in D_{11} is allowed when the inequalities I_6^1, I_{11}^6 and $I_{11} = (0 < \kappa_1/\gamma_1 < \underline{\theta}_{11}) \wedge (\overline{\theta}_{21} < \kappa_3/\gamma_2 < \underline{\theta}_{22})$ hold.

Let us observe, that at present, as we have not yet tackled the problem of identifying the admissible connections between entrance points and exit points, the algorithm may generate spurious behaviors, that is trajectories that can never occur for any set of numerical values of parameters. The behavior QB_2, e.g., is spurious as I_{12}^7 is not consistent with I_{11}^{12}. Similarly, QB_7 and QB_{14} are spurious. However, by checking the consistency of all inequalities that belong to a branch in a BT, we can identify and filter out possible spurious behaviors. As $n = 2$, a representation in the phase plane of the trajectories described by the possibly filtered tree is also given (Fig. 3(a)).

The simulation outcomes are numerically confirmed, and in Fig. 4 we report some of the numerical simulations performed under different conditions. In the following, we give a sketch of the algorithms at work when it calculates $QS(D_1)$, $QS(D_7)$.



(a)

I_3^2	$I_0 \wedge (\frac{\kappa_2}{\gamma_1} > \theta_{11})$
I_7^2	$I_0 \wedge (\frac{\kappa_2}{\gamma_1} < \theta_{11})$
I_5^4, I_5^9	$I_0 \wedge (\frac{\kappa_2}{\gamma_1} > \theta_{12})$
I_9^3, I_4^3	$I_0 \wedge (\frac{\kappa_2}{\gamma_1} > \overline{\theta}_{12})$
I_5	$I_0 \wedge (\overline{\theta}_{12} < \frac{\kappa_2}{\gamma_1} < \overline{x}_1)$
I_{11}^7, I_{11}^{12}	$I_0 \wedge (\frac{\kappa_1}{\gamma_1} < \theta_{11})$
I_{12}^7, I_{12}	$I_0 \wedge (\frac{\kappa_1}{\gamma_1} > \theta_{11})$
I_{11}	$I_0 \wedge (\frac{\kappa_1}{\gamma_1} < \underline{\theta}_{11})$

(b)

Fig. 3. (a) Phase space representation of trajectories described by BT after filtering; • denotes a stable state. (b) Inequalities calculated by the algorithm. I_2^1, I_6^1, I_7^1, I_8^3, I_{11}^6, I_7^8, I_{13}^8, I_{12}^{13}, I_{13}^9 are equal to I_0.

Calculation of $QS(D_1)$. First, the algorithm calculates the set $A(D_1) = \{D_6, D_7, D_2\}$ and the relative positions $V(D_6, D_1) = (0, 1)$, $V(D_7, D_1) = (1, 1)$, $V(D_2, D_1) = (1, 0)$. In D_1 the model (7) reduces to the linear ODEs:

$$\begin{aligned} \dot{x_1} &= \mu_1 - \gamma_1 x_1, \\ \dot{x_2} &= \mu_2 - \gamma_2 x_2, \end{aligned} \tag{9}$$

where $\mu_1 = \kappa_1 + \kappa_2$ and $\mu_2 = \kappa_3$. Transitions from D_1 are possible under the following conditions on parameters:

$$\begin{aligned} &\hat{I}_2^1 : \dot{x_1} > 0 \Rightarrow (\tfrac{\kappa_1+\kappa_2}{\gamma_1} > \overline{\theta}_{11}) \quad \text{to go to } D_2, \\ &\hat{I}_6^1 : \dot{x_2} > 0 \Rightarrow (\tfrac{\kappa_3}{\gamma_2} > \overline{\theta}_{21}) \qquad \text{to go to } D_6, \\ &\hat{I}_7^1 : \dot{x_1} > 0, \dot{x_2} > 0 \Rightarrow \hat{I}_2^1 \wedge \hat{I}_6^1 \text{ to go to } D_7. \end{aligned} \tag{10}$$

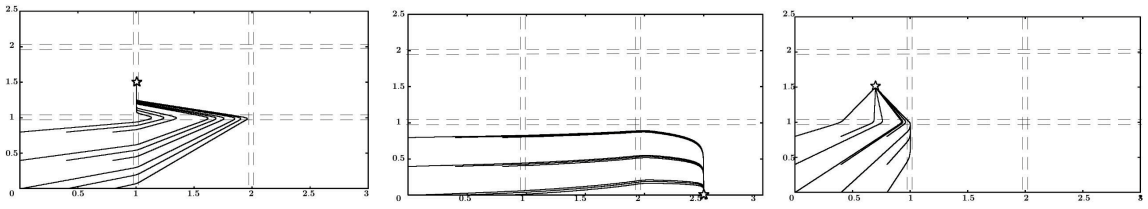
As the parameter space domains defined by $I_2^1 : I_0 \wedge \hat{I}_2^1$, $I_6^1 : I_0 \wedge \hat{I}_6^1$ and $I_7^1 : I_0 \wedge \hat{I}_7^1$ belong to PSD_0, $QS(D_1) = \{D_2, D_6, D_7\}$, and the simulation spawns through three different paths. Let us follow the path related to condition I_7^1 and calculate $QS(D_7)$.

Calculation of $QS(D_7)$. In the switching domain D_7 the Boundary Layer System is:

$$\begin{aligned} Z'_{11} &= \frac{Z_{11}(1 - Z_{11})}{\theta_{11}}(\kappa_1(1 - Z_{11}) + \kappa_2(1 - Z_{21}) - \gamma_1\theta_{11}), \\ Z'_{21} &= \frac{Z_{21}(1 - Z_{21})}{\theta_{21}}(\kappa_3 - \gamma_2\theta_{21}). \end{aligned} \tag{11}$$

The set $\mathcal{F}(\mathcal{Z}(D_7))$ has five elements, the four faces F_k corresponding to D_2, D_6, D_8, D_{12}, and the interior of $\mathcal{Z}(D_7)$. The associated Jacobian matrices are:

$$\mathbf{J_{int(\mathcal{Z}(D_7))}} = \begin{pmatrix} -\kappa_1 & -\kappa_2 \\ 0 & 0 \end{pmatrix}, \ \mathbf{J_2} = (-\kappa_1), \ \mathbf{J_6} = (0), \ \mathbf{J_8} = (0), \ \mathbf{J_{12}} = (-\kappa_2).$$



(a) (b) (c)

Fig. 4. Phase space plots of the numerical simulations performed with different parameter sets and initial conditions taken on an uniform grid of points in D_1. Common parameter values are: $\theta_{11} = \theta_{21} = 1$, $\theta_{12} = \theta_{22} = 2$, $q = 0.01$, $\kappa_3 = 1.5$, $\gamma_2 = 1$. Other parameters are: (a) $\kappa_1 = 2.5, \kappa_2 = 2.5, \gamma_1 = 1$; (b) $\kappa_1 = 25, \kappa_2 = 2.5, \gamma_1 = 10$; (c) $\kappa_1 = 0.7, \kappa_2 = 0.7, \gamma_1 = 1$. QB_4 and QB_{11} abstract the trajectories in (a), QB_9 in (b), and QB_1 and QB_{16} in (c).

Only $\mathbf{J_2}$ and $\mathbf{J_{12}}$ have a complete loop. Then, the algorithm looks for the stationary state on F_2 and F_{12}: $\tilde{\mathbf{Z}}^{\mathbf{2}} = (1 + \kappa_2/\kappa_1 - \gamma_1\theta_{11}/\kappa_1, 0)$ and $\tilde{\mathbf{Z}}^{\mathbf{12}} = (1 - \gamma_1\theta_{11}/\kappa_1, 1)$. The exit point candidate set is built by adding the vertices to these points. Then, the algorithm imposes that both $\tilde{Z}_1^2$ and $\tilde{Z}_1^{12}$ are in (0,1) (line 13 of algorithm 2):

$$0 < \tilde{Z}_1^2 < 1 \Rightarrow (\kappa_1 + \kappa_2 > \gamma_1\theta_{11}) \wedge (\kappa_2 < \gamma_1\theta_{11}), \tag{12}$$

$$0 < \tilde{Z}_1^{12} < 1 \Rightarrow (\kappa_1 > \gamma_1\theta_{11}) \wedge (\gamma_1\theta_{11} > 0). \tag{13}$$

Stability conditions for $\tilde{\mathbf{Z}}^{\mathbf{2}}$ and $\tilde{\mathbf{Z}}^{\mathbf{12}}$ are both fulfilled as $-\kappa_1 < 0$ and $-\kappa_2 < 0$, whereas stability conditions on variable $Z_l, l = 2$ requires that:

$$Z_2'(\tilde{\mathbf{Z}}^{\mathbf{2}}) < 0 \Rightarrow \kappa_3 - \gamma_2\theta_{21} < 0, \tag{14}$$

$$Z_2'(\tilde{\mathbf{Z}}^{\mathbf{12}}) > 0 \Rightarrow \kappa_3 - \gamma_2\theta_{21} > 0. \tag{15}$$

Condition I defined by (15) is compatible with (8), but condition (14) is not. Then, $\tilde{\mathbf{Z}}^{\mathbf{2}}$ is removed from the exit point set, whereas $\tilde{\mathbf{Z}}^{\mathbf{12}}$ is an exit point if $I_{12}^7 : I_0 \wedge I$ holds. Stability conditions on vertices are fulfilled only on vertex $\tilde{\mathbf{Z}}^{\mathbf{11}} = (0, 1)$. Then, the *exit domains* are D_{12}, D_{11}, and $QS(D_7) = \{D_{12}, D_{11}\}$.

5 Discussion and Future Work

The work described above is the first step towards the realization of a qualitative simulation algorithm that aims at (i) generating, from a given initial state, all and none but the trajectories of a class of nonlinear ODE models of GRNs, and (ii) providing the constraints on parameters that should be satisfied so that a specific behavior occurs. At the current stage, the algorithm guarantees that, for $0 < q < \overline{q} \ll 1$, the behavior tree captures all of the sound behaviors. However, as we have not yet performed a thorough analysis with respect to entrance-exit transition, or in other words, we have not yet solved the problem (i) of identifying the only admissible connections between entrance and exit points, the behavior tree may also contains spurious behaviors. Moreover, singular perturbation analysis is a "local" procedure that works quite well in a quantitative context but that needs, in a qualitative context, to be supported by a "global" criterion when local paths are combined to produce a specific trajectory. We are quite confident

that when the solution of (i) together with (ii) has been automatized, the consistency of the whole sequence of inequalities that characterizes a behavior will allow us to filter out all spurious solutions, and to prove the soundness and completeness of our algorithm. The tasks (i) and (ii) raise feasible but non-trivial algorithmic and computational issues and require proper symbolic calculation procedures.

Another important methodological issue that needs to be further studied deals with the stability of stationary points. More precisely, this poses two different problems: one related to the validity of the singular perturbation analysis in the presence of zero real part eigenvalues, and the other one to the determination of an upper bound, $\overline{q}$, of q that guarantees the Jacobian matrix stability. Although not a matter of discussion in this paper, we have already solved the latter problem, but a formal proof that stability but not asymptotic stability in the step function limit does not affect the main conclusions from the singular perturbation analysis is still lacking.

Acknowledgements. This work is carried out within the Interdepartmental CNR-BIOINFORMATICS Project. It was supported in part by the National Programme for Research in Functional Genomics in Norway (FUGE) in the Research Council of Norway (grant no. NFR153302).

References

1. de Jong, H.: Modeling and simulation of genetic regulatory systems: A literature review. J. Computational Biology 9, 67–103 (2002)
2. Glass, L., Kauffman, S.A.: The logical analysis of continuous, nonlinear biochemical control networks. J. Theoretical Biology 39, 103–129 (1973)
3. Kuipers, B.: Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge. MIT Press, Cambridge (1994)
4. de Jong, H., Gouzé, J.L., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulations of genetic regulatory networks using piecewise linear models. Bulletin of Mathematical Biology 66, 301–340 (2004)
5. Filippov, A.F.: Differential equations with discontinuous right hand sides. Kluwer Academic Publishers Group, Dordrecht (1988)
6. Dordan, O., Ironi, L., Panzeri, L.: Some critical remarks on GNA (in preparation)
7. Plahte, E., Kjøglum, S.: Analysis and generic properties of gene regulatory networks with graded response functions. Physica D: Nonlinear Phenomena 201, 150–176 (2005)
8. Gjuvsland, A., Plahte, E., Omholt, S.W.: Threshold-dominated regulation hides genetic variation in gene expression networks. BMC Systems Biology 1 (2007)
9. Plahte, E., Mestl, T., Omholt, S.W.: A methodological basis for description and analysis of systems with complex switch-like interactions. Journal of Mathematical Biology 36, 321–348 (1998)
10. Brazhnik, P., de la Fuente, A., Mendes, P.: Gene networks: how to put the function in genomics. Trends in Biotechnology 20, 467–472 (2002)
11. de Jong, H., Geiselmann, J., Batt, G., Hernandez, C., Page, M.: Qualitative simulation of the initiation of sporulation in *Bacillus subtilis*. Bulletin of Mathematical Biology 66, 261–300 (2004)
12. Ropers, D., de Jong, H., Geiselmann, J.: Mathematical modeling of genetic regulatory networks: Stress response in *Esterichia coli*. In: Fu, P., Latterich, M., Panke, S. (eds.) Systems and Synthetic Biology. Wiley & Sons, Chichester (in press)

13. Ropers, D., de Jong, H., Page, M., Schneider, D., Geiselmann, J.: Qualitative simulation of the carbon starvation response in *Esterichia coli*. BioSystems 84, 124–152 (2006)
14. Veflingstad, S.R., Plahte, E.: Analysis of gene regulatory network models with graded and binary transcriptional responses. Biosystems 90, 323–339 (2007)
15. Holmes, M.: Introduction to Perturbation Methods. Springer, Berlin (1995)
16. Wolfram, S.: The Mathematica Book. Wolfram Media (2003)
17. Gross, J., Yellen, J.: Graph Theory and its Applications. Chapman & Hall/CRC Press, New York (2006)

Property Preservation along Embedding of Biological Regulatory Networks*

Mbarka Mabrouki[1,2], Marc Aiguier[1,2],
Jean-Paul Comet[3], and Pascale Le Gall[1,2]

[1] École Centrale Paris
Laboratoire de Mathématiques Appliquées aux Systèmes (MAS)
Grande Voie des Vignes - F-92295 Châtenay-Malabry
marc.aiguier@ecp.fr
[2] Programme d'Épigénomique
523, Place des Terrasses de l'Agora - F-91025 Evry
{mabrouki,pascale.legall}@epigenomique.genopole.fr
[3] Laboratory I3S, UMR 6070 CNRS/UNSA
Algorithmes-Euclide-B, 2000 route des Lucioles
B.P. 121, F-06903 Sophia-Antipolis
comet@unice.fr

Abstract. In the course of understanding biological regulatory networks (BRN), scientists usually start by studying small BRNs that they believe to be of particular importance to represent a biological function, and then, embed them in the whole network. Such a reduction can lead to neglect relevant regulations and to study a network whose properties can be very different from the properties of this network viewed as a part of the whole. In this paper we study, from a logical point of view, the preservation of properties inherited from small BRNs. The signature of BRN, constituted by a graph, is one of the distinctive features on which embeddings can be defined which leads us to give a first condition on the subgraphs ensuring the preservation of properties of the embedded graphs.

Keywords: Biological regulatory networks, network embedding, property preservation, mathematical modeling, computational tree logic.

1 Introduction

To understand biological regulatory networks (BRN for short), modeling frameworks and simulation technics are often useful since the complexity of the interactions between constituents of the network (mainly genes and proteins) makes intuitive reasoning difficult [3]. Nevertheless, simulation technics are in practice difficult to manage for most of the systems because they are either large, complex or only partially known. Indeed, the lack of precise knowledge about the

* This work is performed within the European project GENNETEC *(GENetic NeTworks: Emergence and Complexity)* STREP 34952.

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 125–138, 2008.

system (are all constituents/interactions taken into account? Which values are given to parameters? Which is the confidence on these parameters?...) is one of the more accurate difficulties to handle computationally all possible hypotheses on the system. Qualitative modeling frameworks have then arose [7,13,5]: they consist in abstracting continuous concentrations of constituents into qualitative ones (discrete and finite) although preserving qualitative observations (like presence/absence of a constituent, increasing of the concentration of a target when increasing the one of a regulator...).

We focus in this paper on the multivalued discrete approach developed by R. Thomas and co-workers [13], in which the concentrations of constituents are abstracted by integers to denote thresholds from which constituents can act on other ones in the network. In this formalism, biological systems are described by an interaction graph defining the static part of the system from which we can build a huge but finite set of state transition graphs defining all the possible dynamics of the system. However, given an interaction graph, just a few dynamic models meet the set of biological experiment observations bringing into play interactions between graph's constituents. To cut down in the class of dynamic models and just preserve the good candidates, some recent works expressed these biological experiment observations by temporal properties and used various model-checking technics to select suitable dynamic models [4,2,9]. From these works, two software tools have been developed: GNA [4] which automatically checks that a given dynamic model satisfies some biological experiment observations, and SMBioNet [2] which cuts down in the whole class of dynamic models to select the ones that satisfy some given biological experiment observations. In both cited tools, temporal properties denoting biological experiment observations have been expressed in Computation Tree Logic [6].

These logical approaches based on model-checking technics have been shown very efficient to study small BRNs but are not well-adapted for large BRNs. The well-known reason is because model-checking technics are time consuming. Indeed, we have to deal with the limit given by complexity theory: model-checking is based on NP-hard or exponential algorithms. In practice, human cleverness is used to find the situations for which model-checking may become tractable. This is what we propose to do in this paper. Indeed, BRNs are generally embedded into other ones. To allow to describe and study BRN behaviors in the large, we propose in this paper to study the consequences of the embedding. More precisely, we propose to study which are the conditions to impose on the embedding to preserve the dynamics of sub-BRNs. We show that, generally speaking, questioning temporal properties (i.e. biological experimental observations) leads us to study the dynamics of the global BRN "from scratch", i.e without taking benefit of the dynamics of the sub-BRNs, which can be unacceptable in running time. On the contrary, if all dynamics of sub-BRNs are preserved, this then leads us just to focus on the biological experiment observations linked with interactions of the sub-BRNs between them. We can then hope to be able to use both previous tools to automatically study the dynamics of the global BRN. Moreover, this approach corresponds to the classical method used by most of

biologists when they study a biological system. They start by studying small BRNs that they believe to be of particular importance to represent a biological function. The interactions of this BRN with the external genes, are studied only afterwards even if these external genes potentially could influence the behavior of the studied part. Of course, this bottom-up approach makes sense only if there is a complete preservation of sub-systems behaviors as this has been done in this paper up to some sufficient conditions. Otherwise, systems can only be studied globally because of the apparition of emergent properties. To comprehend this notion of emergent properties, we introduced in [1] an abstract mathematical denotation for complex systems.

The paper is then structured as follows: after some reminders on the temporal logic CTL in Section 2, Section 3 presents a logical characterization for BRNs. Section 4 presents the main result of this paper: the preservation of properties through the embedding of BRNs into larger networks. In Section 5, we give a counter-example to justify the constraints we put on graph embedding to ensure property preservation. Finally in Section 6 we give some concluding remarks.

Let us notice the particularity of the logic for BRN presented in this paper: signatures are not simple sets of symbols but are interaction graphs (the static part of BRN). This is what makes tough the definition of the embedding (see Definition 2) as well as the definitions of the consequences of the embedding both on biological experiment observations expressed over sub-BRNs (see Definition 4) and on the dynamics of sub-BRNs embedded into a larger one (see Definition 8). This is what makes also nontrivial the proof of the preservation of temporal properties through embedding.

2 Preliminaries

Computational tree logic (CTL) [6] is a branching-time temporal logic where the structure representing all possible executions is *tree-like* rather than linear. It is well-adapted to specify and reason about non-deterministic and/or concurrent processes. Here, we consider actually a restriction of CTL by removing the next operator X, noted CTL-X [14,15]. The reason is for biological applications, the logical connector X is not of big relevance. The reason is twofold. First, the time mandatory for a biological system to change of qualitative state is not deterministic and the elapsed time between two consecutive states has a large variance. Secondly, the discretization of the dynamical system abstracts the quantitative time (represented by $t \in \mathbb{R}^+$) into a qualitative time ($n \in \mathbb{N}$). Then the real time necessary for a NEXT transition of the biological system depends also on the number of intermediate states used for the discretization step.

When dealing with propositional fragment of logics, a signature $Atom$ is only a set of propositional variables which are the atomic formulas.

Given a signature $Atom$, a model over $Atom$, so-called Kripke frame, is a transition system (S, T) where:

- S is a set whose elements are usually called *states*;
- $T \subseteq S \times S$ is a binary relation satisfying: $\forall s \in S, \exists s' \in S,\ (s, s') \in T$;

and (S, T) is equipped with a total function $L : S \rightarrow 2^{Atom}$ called *labeling function*.

Therefore, models over *Atom* are labeled transition systems where T denotes the transition relation and L is the labeling associating for each state s of S the set of propositional variables true at s.

Formulas over *Atom* are well-formed formulas whose syntactical rules are given by:

$$\begin{aligned} For ::= {} & ATOM \mid For \Rightarrow For \mid For \wedge For \mid For \vee For \mid \neg For \\ & AG\ For \mid EG\ For \mid AF\ For \mid EF\ For \mid A[For\ U\ For] \mid E[For\ U\ For] \end{aligned}$$

The intuitive meaning of modal operator $F\varphi$ (resp. $G\varphi$) means that φ will be finally (F) (resp. is globally (G)) true. The prefix A (resp. E) means that the formula is true for all possible futures (resp. there exists a future for which the following property is true). Finally, formulas of the form $\varphi\ U\ \psi$ mean that φ has to be true until (U) ψ becomes true. They are also preceded by the prefixes A or E.

The validity of formulas is expressed *via* a binary relation usually denoted by $\models$ between models and formulas over a set of atomic formulas *Atom*. A path is any sequence $\sigma = (s_0, s_1, \ldots, s_n, \ldots)$ such that for every $i \in \mathbb{N}$ we have $(s_i, s_{i+1}) \in T$. Then, $(S, T) \models \varphi$ if for any state $s \in S$, (S, T) *satisfies* φ, denoted by $((S, T), s) \models \varphi$, according to the following inductive definition:

- $((S, T), s) \models p$ iff $p \in L(s)$ for $p \in Atom$;
- $((S, T), s) \models AG\varphi$ (resp. $((S, T), s) \models EG\varphi$) iff for every (resp. there exists a) path $(s_0, s_1, \ldots, s_n, \ldots)$, for every $i \in \mathbb{N}$, $((S, T), s_i) \models \varphi$;
- $((S, T), s) \models AF\varphi$ (resp. $((S, T), s) \models EF\varphi$) iff for every (resp. there exists a) path $(s_0, s_1, \ldots, s_n, \ldots)$, there exists $i \in \mathbb{N}$, $((S, T), s_i) \models \varphi$;
- $((S, T), s) \models A[\varphi\ U\ \psi]$ (resp. $((S, T), s) \models E[\varphi\ U\ \psi]$) iff for every (resp. there exists a) path $(s_0, s_1, \ldots, s_n, \ldots)$, there exists $i \in \mathbb{N}$ such that $((S, T), s_i) \models \psi$ and for every $j < i$, $((S, T), s_j) \models \varphi$;
- Boolean connectives are handled as usual.

In the sequel, to prove the preservation of properties through the embedding of biological regulatory networks, we will use a standard equivalence relation on the states of transition systems, the so-called *divergence blind stuttering equivalence* (dbs), which have been proved to preserve CTL-X formulas, i.e. the transition system and its quotient, with respect to the dbs equivalence relation, are elementary equivalent [10].

Let us recall the definition of a dbs relation R on a transition system (S, T).

A binary relation R on S is called a *divergence blind stuttering* (dbs) relation if, and only if it is symmetric and

$$r\ R\ s \Longleftrightarrow \begin{cases} L(r) = L(s) \\ (r, r') \in T \Rightarrow \exists s_0, s_1, \ldots, s_n \text{ finite path }, n \geq 0, (s_0 = s) \\ \qquad\qquad \wedge (\forall i < n, r\ R\ s_i) \wedge r'\ R\ s_n \end{cases}$$

It is obvious to show that every dbs relation is transitive. Moreover, as the case $n = 0$ is allowed in the second condition, the empty relation is a dbs relation. Finally, the diagonale relation on S is also a dbs relation, and it is easy to show that dbs relations are closed under union. Hence, the largest dbs relation exists and is an equivalence relation noted $\simeq_{dbs}$.

Given a transition system (S, T), its quotient by $\simeq_{dbs}$, denoted $(S, T)_{/\simeq_{dbs}}$, is defined by:

- the set of states $S_{/\simeq_{dbs}}$ is the set of equivalence classes of $\simeq_{dbs}$, $[s]$ denoting the equivalence class of s for s state of S
- the set of transitions $T_{/\simeq_{dbs}}$ defined by $([s], [t]) \in T_{/\simeq_{dbs}}$ iff there exists $s' \in [s]$ and $t' \in [t]$ such that $(s, t) \in T$
- $(S, T)_{/\simeq_{dbs}}$ is provided with the labeling function $L_{/\simeq_{dbs}}$ defined by $L_{/\simeq_{dbs}}([s]) = L(s)$.

3 BRN Logic

In this section, we will present the multivalued discrete approach developed by R. Thomas [13] for genetic regulatory networks as a logic built over the logic CTL-X. We will follow the standard approach for presenting a logic, *i.e.* syntax (signatures and formulas) and semantics (models and the satisfaction relation).

3.1 Syntax

Signatures. A biological regulatory network is represented by a labeled directed graph, called interaction graph. Vertices abstract biological entities, as genes or proteins, and will be called *variables*. Edges abstract interactions between variables. When a variable i activates a variable j, variable i can act positively on j, then there exists an edge from i to j labeled by the sign "+". On the contrary, when a variable i inhibits a variable j, variable i can act negatively on j, then there exists an edge from i to j labeled by the sign "-". Moreover, the action, activation or inhibition, between two variables becomes efficient only when the level of concentration of the regulator reaches a given threshold. In the discrete modeling framework of R. Thomas, the concentration levels for the variable i can take a finite number of values $\{0, 1, \ldots, b_i\}$ and thresholds related to the actions of i are numbered from 1 to b_i: the action of i on j is triggered only if the concentration of i crosses its concentration level. Thus, each interaction $i \longrightarrow j$ is labeled by a sign and a threshold. The knowledge of interactions between variables, including signs and thresholds, is called the static part of BRNs and constitutes the elements of signatures for a logic dedicated to BRNs.

Definition 1 (Signature). *A BRN-signature is a labeled directed graph* $G = \langle V, F, Sn, Th \rangle$ *where:*

1. V *is a finite set whose the elements are called* variables.
2. $F \subseteq V \times V$ *denotes the set of edges.*
3. Sn *is a mapping from* F *to* $\{+, -\}$.

4. Th is a mapping from F to $\mathbb{N}^*$ such that:

$$\forall (i,j) \in F \ \forall l \in \mathbb{N}^* \ \exists k \in V \ (Th(i,j) = l \ \wedge \ l \neq 1 \Rightarrow (i,k) \in F \wedge Th(i,k) = l-1)$$

Point 4. gives some restrictions on the way the edges are labeled. If an edge outgoing from a variable i is labeled by $l \geq 2$, then there exist edges outgoing from i labeled by $1, \ldots, l-1$. This well represents the qualitative nature of thresholds in BRN as used in this paper.

Notation 1. *Let $G = \langle V, F, Sn, Th \rangle$ be a BRN-signature and i be a variable in V. G_i^+, resp. G_i^-, denotes the set of successors, resp. predecessors, of i in G, and b_i denotes the cardinal of the set of thresholds for i. Formally, we have:*

$$G_i^+ = \{j \in V | (i,j) \in F\}$$

$$G_i^- = \{j \in V | (j,i) \in F\}$$

$$b_i = |\{l \in \mathbb{N}^* \mid \exists j \in G_i^+, Th(i,j) = l\}|$$

Example 1. *To illustrate Definition 1, we take as running example a model inspired from the one of control of immunity in temperate bacteriophage lambda. This model, proposed by Thieffry and Thomas in [11], contains genes cI and cro: cI inhibits cro and activates its own synthesis whereas the variable cro inhibits the expression of both variables, see Figure 1. The associated BRN-signature, denoted G_1 in the sequel, is simply given by:*

$$\begin{array}{l} \langle \ \{cI, cro\}, \{(cI,cI),(cI,cro),(cro,cI),(cro,cro)\}, \\ \quad Sn : \{(cI,cI) \mapsto +, (cI,cro) \mapsto -, (cro,cI) \mapsto -, (cro,cro) \mapsto -\}, \\ \quad Th : \{(cI,cI) \mapsto 1, (cI,cro) \mapsto 1, (cro,cI) \mapsto 1, (cro,cro) \mapsto 2\} \ \rangle \end{array}$$

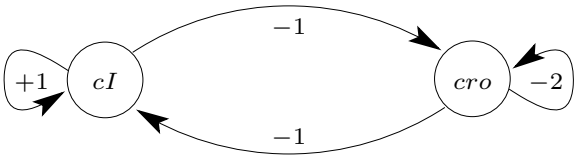


Fig. 1. Interaction graph for the $cI - cro$ system

Signature embedding. Biologists can identify small parts issued from a BRN involving a large number of genes. These parts are assimilated to a biological function insofar as it can be proven that the biological function is essentially related to the concentration levels of the variables occurring in the considered subpart.

Embedding of BRN signatures can formalize such an approach. However, signature embedding cannot be simple graph embeddings (which is defined by Conditions 1 and 2 of Definition 2 just below). Indeed, as well as preserving edge signs (see Condition 2), as the thresholds on edges depend on the properties of the graph (a threshold cannot be greater than the number of outgoing edges), it matters to pay attention to the preservation of the conditions on the thresholds (Conditions 3 and 4). In fact, as thresholds are taken into consideration

in signatures, the key point to carry through the embedding is the preservation of the equality between thresholds and the numerical order between them. New intermediate thresholds for a given variable can be introduced when including a BRN in another one, but relationships between existing thresholds have to be preserved in the larger one. Finally, a supplementary condition (Condition 5) has to be added. This condition means the preservation of predecessors in interaction graphs. This condition can seem very restrictive. However, it is useful to ensure the preservation of properties inherited from the small BRN to the large BRN (see the counter-example given in Section 5) which makes fail the preservation when Condition 5 does not hold. This leads to the following definition:

Definition 2 (Signature Embedding). *Let G and G' be BRN-signatures such that $G = \langle V, F, Sn, Th\rangle$ and $G' = \langle V', F', Sn', Th'\rangle$. A signature embedding $G \to G'$ is an injective mapping $\sigma : V \to V'$ such that:*

1. $\forall i, j \in V, (i,j) \in F \Leftrightarrow (\sigma(i), \sigma(j)) \in F'$
2. $\forall i, j \in V, (i,j) \in F, Sn(i,j) = Sn'(\sigma(i), \sigma(j))$
3. $\forall i \in V, \forall j, k \in G_i^+, Th(i,j) = Th(i,k) \Leftrightarrow Th'(\sigma(i), \sigma(j)) = Th'(\sigma(i), \sigma(k))$
4. $\forall i \in V, \forall j, k \in G_i^+, Th(i,j) < Th(i,k) \Leftrightarrow Th'(\sigma(i), \sigma(j)) < Th'(\sigma(i), \sigma(k))$
5. $\forall j \in V,\ \forall k' \in V', (k', \sigma(j)) \in F' \Rightarrow \exists i \in V, (i,j) \in F \wedge \sigma(i) = k'$.

Notation 2. *Let $\sigma : G \to G'$ be a signature embedding where $G = \langle V, F, Sn, Th\rangle$ and $G' = \langle V', F', Sn', Th'\rangle$ and let ω a set of variables in V, $\sigma(\omega)$ denotes the set $\{\sigma(i) \mid i \in \omega\}$.*

Example 2. *Figure 2 presents the BRN-signature G_2, sharing with G_1 both variables cI and cro, and containing a new variable N. According to Definition 1, a signature embedding σ_{id} between $\{cI, cro\}$ and $\{cI, cro, N\}$ can be defined: $\sigma_{id}(cI) = cI$ and $\sigma_{id}(cro) = cro$. Conditions 1 and 2 are clearly verified (all edges of G_1 are in G_2 labeled with the same sign). Condition 3 requires that the equality between thresholds for outgoing edges in G_1 is preserved in G_2, it is verified since only $Th(cI, cI) = Th(cI, cro)$ in G_1 and we have $Th'(cI, cI) = Th'(cI, cro)$ in G_2. Condition 4, which requires that the order between thresholds for outgoing edges in G_1 is preserved in G_2, is also verified. For instance, in G_1, cro has two outgoing edges (cro, cI) and (cro, cro) with $Th(cro, cI) < Th(cro, cro)$. In G_2, we have $Th'(cro, cI) < Th'(cro, cro)$. Condition 5 is also verified (cI and cro in G_2 have no new predecessors with respect to G_1).*

Roughly speaking, we can link two BRN-signatures by a signature embedding when the addition of new variables has only the effect of shifting the thresholds issued from the inherited variables.

Formulas. Formulas for BRN are simply CTL-X formulas whose atomic formulas describe comparisons between a concentration level of a variable with some threshold values.

Definition 3 (BRN Formulas). *Let $G = \langle V, F, Sn, Th\rangle$ be a BRN-signature. Formulas over G are CTL-X formulas whose atomic formulas are of the form $(i \sim l)$ where $i \in V$, $l \in \{0, \ldots, b_i\}$ and $\sim \in \{=, <, >\}$.*

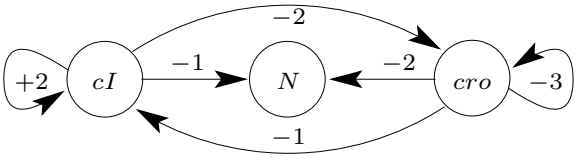


Fig. 2. BRN-signature G_2

We denote by $Atom(G)$ the set of atomic formulas built on G and by $Sen(G)$ the set of formulas over G.

In the sequel, $i \geq l$ (resp. $i \leq l$) will denote the formula $i = l \vee i > l$ (resp. $i = l \vee i < l$).

Signature embeddings obviously rename variables and thresholds occurring in atomic formulas. However, the threshold renaming is not so simple. Indeed, the presence of new variables makes side effects on the thresholds by shifting them. This gives rise to the following definition:

Definition 4 (Formula Renaming). *Let $\sigma : G \to G'$ be a signature embedding with $G = \langle V, F, Sn, Th\rangle$ and $G' = \langle V', F', Sn', Th'\rangle$. For all $i \in V$,let us note $\sigma_i : \{0, 1, \ldots, b_i\} \to \{0, 1, \ldots, b_{\sigma(i)}\}$ the mapping defined by:*

- $\sigma_i(0) = 0$
- *For all $l \neq 0$, $\sigma_i(l) = Th'(\sigma(i), \sigma(j))$ with j any arbitrary variable such that $j \in G_i^+$ and $Th(i, j) = l$*

Let us note $\overline{\sigma} : Atom(G) \to Sen(G')$ the mapping defined by:

- *For all $(i = l) \in Atom(G)$ with $l \neq b_i$, $\overline{\sigma}(i = l) = \sigma(i) \geq \sigma_i(l) \wedge \sigma(i) < \sigma_i(l+1)$*
- *For all $(i = b_i) \in Atom(G)$, $\overline{\sigma}(i = b_i) = \sigma(i) \geq \sigma_i(b_i)$*
- *For all $(i > l) \in Atom(G)$, $\overline{\sigma}(i > l) = \sigma(i) \geq \sigma_i(l + 1)$*
- *For all $(i < l) \in Atom(G)$, $\overline{\sigma}(i < l) = \sigma(i) < \sigma_i(l)$*

Let us note $\sigma^\sharp$ the canonical extension of the signature embedding σ on formulas in $Sen(G)$ defined as follows:

- *For $p \in Atom(G)$, $\sigma^\sharp(p) = \overline{\sigma}(p)$,*
- *For other formulas, Boolean connectives and temporal operators are preserved.*

The definition explains how to convert formulas in $Sen(G)$ into formulas in $Sen(G')$ by following the simple idea of translating a threshold into an interval of possible values.

3.2 Semantics

Models. Each variable i in a BRN-signature G is a genetic entity which is characterized at a given point in time by a concentration level. Dealing with regulatory networks with thresholds whose the set of nodes is finite, the state space generated from G is finite and defined by:

Definition 5 (State). *Let $G = \langle V, F, Sn, Th\rangle$ be a BRN-signature. The state space S_G of G is the set of mappings $s : V \to \mathbb{N}$ such that for every $i \in V$, $s(i) \in \{0, \ldots, b_i\}$.*

Example 3. *In the BRN-signature $G1$ of Example 1, Variables cI and cro have, respectively 2 and 3 possible concentration levels: 0 or 1, and 0, 1 or 2. Therefore, The state space for G_1 is $S_{G_1} = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2)\}$.*

The concentration level of each variable $i \in V$ of a given BRN-signature G, evolves over time depending on the concentration level of its resources (*i.e.* sets of i's predecessors in G which have reached a concentration level to affect i's one by making it increase or decrease). However, neither G nor the concentration level of i's resources gives clues to decide the concentration level that i can reach. This is a degree of freedom of BRN-signatures which gives rise to a class of possible G-models, so-called dynamics of G. All these possible G-models do not correspond to actual biological functions. This is by biological knowledge described by CTL-X properties that we can cut down in the class of all possible G-models. Formally, G-models are defined as follows:

Definition 6 (Resources). *Let G be a BRN-signature. The set of resources $R_{G,i}(s)$ of a variable i at the state $s \in S_G$ is defined by:*

$$R_{G,i}(s) = \begin{cases} \{j \in G_i^- | (Sn(j,i) = + \text{ and } s(j) \geq Th(j,i))\} \\ \cup \\ \{j \in G_i^- | (Sn(j,i) = - \text{ and } s(j) < Th(j,i))\} \end{cases}$$

Hence, a resource is the presence of an activator or the absence of an inhibitor.

Example 4. *Figure 3 gives the sets of resources for the three variables cI, cro and N in S_{G_1} and S_{G_2}.*

cI	cro	$R_{G,cI}$	$R_{G,cro}$
0	0	$\{cro\}$	$\{cI, cro\}$
0	1	$\emptyset$	$\{cI, cro\}$
0	2	$\emptyset$	$\{cI\}$
1	0	$\{cI, cro\}$	$\{cro\}$
1	1	$\{cI\}$	$\{cro\}$
1	2	$\{cI\}$	$\emptyset$

cI	cro	N	$R_{G',cI}$	$R_{G',cro}$	$R_{G',N}$
0	0	0	$\{cro\}$	$\{cI, cro\}$	$\{cI, cro\}$
0	1	0	$\emptyset$	$\{cI, cro\}$	$\{cI, cro\}$
0	2	0	$\emptyset$	$\{cI, cro\}$	$\{cI\}$
0	3	0	$\emptyset$	$\{cI\}$	$\{cI\}$
1	0	0	$\{cro\}$	$\{cI, cro\}$	$\{cro\}$
1	1	0	$\emptyset$	$\{cI, cro\}$	$\{cro\}$
1	2	0	$\emptyset$	$\{cI, cro\}$	$\emptyset$
1	3	0	$\emptyset$	$\{cI\}$	$\emptyset$
2	0	0	$\{cI, cro\}$	$\{cro\}$	$\{cro\}$
2	1	0	$\{cI\}$	$\{cro\}$	$\{cro\}$
2	2	0	$\{cI\}$	$\{cro\}$	$\emptyset$
2	3	0	$\{cI\}$	$\emptyset$	$\emptyset$

Fig. 3. Resources of cI, cro and N in S_{G_1} (left) and in S_{G_2} (right)

Definition 7 (G-models). *Let $G = \langle V, F, Sn, Th\rangle$ be a BRN-signature and let $\kappa = \{(i, w) \mid i \in V \wedge w \subseteq G_i^-\}$ be the set of all subsets of predecessors in G for every variable $i \in V$. A G-model is a mapping $p : \kappa \to \mathbb{N}$ such that: $\forall (i, w) \in \kappa,\ p((i, w)) \in \{0, \ldots, b_i\}$.*

Example 5. *From the BRN-signature G_2 of Figure 2, we have the following set κ:*

$$\kappa = \begin{cases} \{(cI, \emptyset), (cI, \{cI\}), (cI, \{cro\}), (cI, \{cI, cro\})\} \\ \cup \\ \{(cro, \emptyset), (cro, \{cI\}), (cro, \{cro\}), (cro, \{cI, cro\})\} \\ \cup \\ \{(N, \emptyset), (N, \{cI\}), (N, \{cro\}), (N, \{cI, cro\})\} \end{cases}$$

From the value of the concentration levels for cI, cro and N, a possible G_2-model p_2 is given in Figure 4 (left).

Signature embeddings $\sigma : G \to G'$ have a counterpart on models which is expressed by a classic forgetful mapping. Here also, some difficulties occur due to some restrictions to make on thresholds from the "richer" model defined on G' to the "poorer" one defined on G. This then leads to the following definition:

Definition 8 (Reduced model). *Given a signature embedding $\sigma : G \to G'$ and a G'-model p', the reduced G-model p from p', denoted $p'_{|_\sigma}$, is defined as follows: $\forall (i, w) \in \kappa$,*

$$p((i, w)) = \begin{cases} Th(i, j) \text{ if it exists } j \text{ in } V \text{ such that} \\ \qquad Th'(\sigma(i), \sigma(j)) = max_{(i,k)\in F}\{Th'(\sigma(i), \sigma(k)) \mid \\ \qquad\qquad Th'(\sigma(i), \sigma(k)) \leq p'((\sigma(i), \sigma(w)))\} \\ 0 \qquad \text{otherwise} \end{cases}$$

Example 6. *Figure 4 (right) gives the reduced G_1-model p_1 of p_2 along the signature embedding given in Example 2.*

From a G-model p, a transition system (S_G, T) can be generated where the transitions in T give the state evolution as described in p. Here, two possibilities can occur. We make evolve either many variables directly to their concentration level specified by p, or one variable i and only by one unit in the direction of $p((i, \omega))$ where ω is the set of resources of i at the current state. These two possibilities are respectively called *synchronous* and *asynchronous* description of the G-model p. Here, we follow the asynchronous description because in the nature, it is unlikely that, *in vivo*, several variables cross a threshold simultaneously [12].

resource ω'	$p_2((cI, \omega'))$	$p_2((cro, \omega'))$	$p_2((N, \omega'))$
$\emptyset$	0	0	0
$\{cI\}$	2	2	0
$\{cro\}$	2	1	0
$\{cI,cro\}$	2	3	0

resource ω	$p_1((cI, \omega))$	$p_1((cro, \omega))$
$\emptyset$	0	0
$\{cI\}$	1	1
$\{cro\}$	1	1
$\{cI,cro\}$	1	2

Fig. 4. A G_2-model p_2 (left) and its reduced G_1-model p_1 (right)

Definition 9 (Asynchronous Transition System). *Let $G = \langle V, F, Sn, Th\rangle$ be a BRN-signature and let p be a G-model. The* asynchronous transition system *generated from p is a directed graph $GTA((G,p)) = (S_G, T)$ such that:*

- $\forall s \in S_G, (s,s) \in T \Leftrightarrow \forall i \in V, s(i) = p((i, R_{G,i}(s)))$
- $\forall s \neq s' \in S_G$, $(s,s') \in T$ *if, and only if:*
  - *there exists $i \in V$, such that*

$$s'(i) = \begin{cases} s(i) \ + \ 1 \text{ and } s(i) < p((i, R_{G,i}(s))) \\ s(i) \ - \ 1 \text{ and } s(i) > p((i, R_{G,i}(s))) \end{cases}$$

  - *and $s'(j) = s(j)$ for every $j \in V \setminus \{i\}$.*

Example 7. *Figure 5 gives from the left to the right, the asynchronous transition systems $GTA((G_1,p_1))$ and $GTA((G_2,p_2))$ generated from p_1 and p_2.*

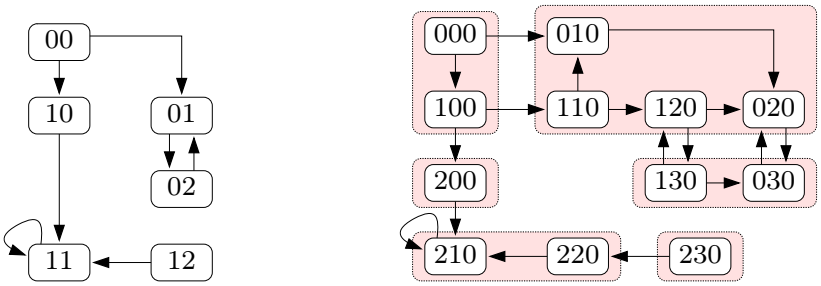


Fig. 5. Asynchronous transition systems $GTA((G_1,p_1))$ and $GTA((G_2,p_2))$. Colored boxes represent the $\simeq_{dbs}$ equivalence classes of $GTA((G_2,p_2))$ – see Section 4.

Satisfaction relation. The asynchronous transition system (S_G, T) generated from a G-model p is a transition system following the definition in Section 2. However, to satisfy CTL formulas, we have to manipulate Kripke frames and then we need to precise the labeling function $L : S_G \to 2^{Atom(G)}$. Given a state s in S_G,

$$L(s) = \{i > l, i < l', i = l'' \mid i \in V, l \in \{0,1,\ldots,b_i - 1\}, l' \in \{1,2,\ldots,b_i\},$$

$$l'' \in \{0,1,\ldots,b_i\}, s(i) > l, s(i) < l', s(i) = l''\}$$

Therefore, the satisfaction relation of a formula φ over a BRN-signature G for a G-model p is then defined by: $p \models \varphi \Longleftrightarrow GTA((G,p)) \models \varphi$ following the definitions given in Section 2.

4 Property Preservation along Signature Embeddings

In this section, we show that given a signature embedding $\sigma : G \to G'$ and a G'-model p', p' and $p'_{|_\sigma}$ are elementary equivalent on formulas in $Sen(G)$ up to σ. This is stated by the following result.

Theorem 1. *For every signature embedding $\sigma : G \to G'$, for every G'-model p' and for every formula $\varphi \in Sen(G)$,*

$$p' \models \sigma^{\sharp}(\varphi) \quad \Longleftrightarrow \quad p'_{|\sigma} \models \varphi$$

Proof (Sketch). Let us consider a signature embedding $\sigma : G \to G'$, a G'-model p' for the BRN-signature G', its associated asynchronous transition system $(S_{G'}, T') = GTA((G', p'))$ and a formula $\varphi \in Sen(G)$. Let us note $(S_G, T) = GTA(G, p'_{|\sigma})$. Start by defining the mapping $B : S_G \to 2^{S_{G'}}$ as follows: for every $s \in S_G$, $B(s)$ is the set of states s' in $S_{G'}$ verifying for every i in V:

- if $s(i) = b_i$, then
$$s'(\sigma(i)) \geq \sigma_i(b_i)$$
- else,
$$s'(\sigma(i)) \geq \sigma_i(s(i)) \wedge s'(\sigma(i)) < \sigma_i(s(i)+1)$$

The proof of Theorem 1 rests on the following intermediate propositions. The proofs of these propositions can be found in [1].

Proposition 1. *The mapping B makes a partition of $S_{G'}$, i.e.*

1. $\forall s, s' \in S,\ B(s) \cap B(s') = \emptyset$, *and*
2. $\bigcup_{s \in S_G} B_s = S_{G'}$.

Note $P = \{B(s) | s \in S_G\}$. Then, we have:

Proposition 2. *P is a dbs equivalence.*

It then remains to prove:

Proposition 3. *$(S_{G'}, T')_{/\simeq_{dbs}}$ and (S_G, T) are isomorphic.*

It is well known that isomorphic models are elementary equivalent (i.e. they satisfy the same set of properties). Therefore, by applying the result of [10], we can conclude that $GTA((G', p'))$ and $GTA((G, p))$ are elementary equivalent on formulas over G up to formula renaming resulting from σ.

Some readers will recognize the so-called satisfaction condition of the institution framework [8]. Hence, this BRN-logic based on signature embeddings is then an institution.

Example 8. *For the current example, the equivalence classes of $\simeq_{dbs}$ in G_2 are highlighted in Figure 5 by colored boxes.*

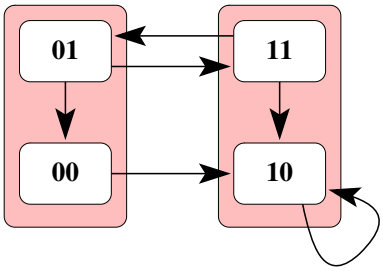
5 Counter-Example Justifying Our Restrictive Notion of Signature Embeddings

In this section we give a counter-example to show the significance of Condition 5 (preservation of predecessors) of Definition 2. Let us consider both BRN-signatures G and G' of figure 6. It is possible to construct an injective mapping

Fig. 6. Counter-example: we consider an embedding not satisfying Condition 5 in Definition 2

resource ω'	$p'((a,\omega'))$	$p'((b,\omega'))$
$\emptyset$	0	0
$\{a\}$	1	
$\{b\}$	1	
$\{a,b\}$	1	

resource ω	$p((a,\omega))$
$\emptyset$	0
$\{a\}$	1



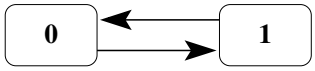


Fig. 7. A G'-model p' and its reduced G-model p

$\sigma : V \longrightarrow V'$ satisfying Conditions 1, 2, 3, and 4 of Definition 2 with $\sigma(a) = a$. For the signature G' we consider the model p' given in Figure 7 (left) from which we deduce the reduced G-model $p = p'_{|_\sigma}$ from p' (see Figure 7-right), and we consider the asynchronous transition systems generated from p and p'.

It is then easy to see that models p' and p do not satisfy the same formulas of CTL-X. For example the formula $AG(AF(a = 0))$ which means that the system will infinitely often pass through a state where $a = 0$ is true, is satisfied by p but not by p'.

6 Conclusion

We have presented the multivalued discrete approach for biological regulatory networks under the classical form of a logical formalism. BRN-signatures are made of graphs, denoting the static part of BRN. Formulas are CTL-X formulas over atoms expressing comparisons between concentration levels of gene products with some abstract discrete values. Models are asynchronous transition systems deduced from the knowledge of parametrization explicating towards which concentration level tends a variable when it is under the influence of other ones. Lastly, the satisfaction relation is simply deduced from the one defined for the CTL-X formalism. In order to study how properties expressed on a small BRN are preserved or not when embedding it within a larger one, we have equipped our BRN formalism with signature embeddings. Their main particularity is that they capture the fact that a concentration level or threshold relative to a network

is converted into an interval of concentration levels. We have proved that CTL-X properties are preserved along such signature embeddings.

We plan to pursue our work by investigating some other conditions which will allow us to go further in the property preservation while keeping a meaning for biological experts. In particular, we are currently investigating under which weaker embedding condition, a weaker set of CTL properties can be still preserved.

References

1. Aiguier, M., Le Gall, P., Mabrouki, M.: A formal denotation of complex systems: how to use algebraic refinement to deal with complexity of systems, Tech. report (2008), `http://www.epigenomique.genopole.fr/~aiguier`
2. Bernot, G., Comet, J.-P., Richard, A., Guespin, J.: Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic. Journal of Theoretical Biology 229(3), 339–347 (2004)
3. de Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. J. Comput. Biol. 9(1), 67–103 (2002)
4. de Jong, H., Geiselmann, J., Hernandez, C., Page, M.: Genetic network analyzer: qualitative simulation of genetic regulatory networks. Bioinformatics 19(3), 336–344 (2003)
5. de Jong, H., Gouzé, J.-L., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulation of genetic regulatory networks using piecewise-linear models. Bulletin of Mathematical Biology 66(2), 301–340 (2004)
6. Emerson, E.A.: Temporal and modal logic. In: Handbook of theoretical computer science formal models and semantics, vol. b, pp. 995–1072. MIT Press, Cambridge (1990)
7. Glass, L., Kauffman, S.A.: The logical analysis of continuous non-linear biochemical control networks. J. Theor. Biol. 39, 103–129 (1973)
8. Goguen, J.A., Burstall, R.-M.: Institutions: Abstract model theory for specification and programming. Journal of the ACM 39(1), 95–146 (1992)
9. Mateus, D., Gallois, J.-P., Comet, J.-P., Le Gall, P.: Symbolic modeling of genetic regulatory networks. Journal of Bioinformatics and Computational Biology (2007)
10. De Nicola, R., Vaandrager, F.: Three logics for branching bisimulation. J. ACM 42(2), 458–487 (1995)
11. Thieffry, D., Thomas, R.: Dynamical behaviour of biological regulatory networks - ii. immunity control in bacteriophage lambda. Bull. Math. Biol. 57(2), 277–297 (1995)
12. Thomas, R.: Logical analysis of systems comprising feedback loops. J. Theor. Biol. 73(4), 631–656 (1978)
13. Thomas, R., d'Ari, R.: Biological feedback. CRC Press, Boca Raton (1990)
14. van Glabbeck, R., Weijland, W.P.: Refinement in branching time semantics. In: Proc. IFIP Conference, pp. 613–618 (1989)
15. Wehrheim, H.: Inheritance of temporal logic properties. In: Najm, E., Nestmann, U., Stevens, P. (eds.) FMOODS 2003. LNCS, vol. 2884, pp. 79–93. Springer, Heidelberg (2003)

Process Algebra Models of Population Dynamics

Chris McCaig, Rachel Norman, and Carron Shankland

Department of Computing Science and Mathematics,
University of Stirling, Stirling, FK9 4LA, UK
{cmc,ran,ces}@cs.stir.ac.uk

Abstract. It is well understood that populations cannot grow without bound and that it is competition between individuals for resources which restricts growth. Despite centuries of interest, the question of how best to model density dependent population growth still has no definitive answer. We address this question here through a number of individual based models of populations expressed using the process algebra WSCCS. The advantage of these models is that they can be explicitly based on observations of individual interactions. From our probabilistic models we derive equations expressing overall population dynamics, using a formal and rigorous rewriting based method. These equations are easily compared with the traditionally used deterministic Ordinary Differential Equation models and allow evaluation of those ODE models, challenging their assumptions about system dynamics. Further, the approach is applied to epidemiology, combining population growth with disease spread.

1 Introduction

The idea that populations cannot grow without bound has been of interest to modellers for centuries. Malthus [1], in 1798, proposed a simple exponential growth model based on compound interest but noted that this was unrealistic, since when a population becomes very large, access to resources will become restricted, restricting further growth in the population. Verhulst proposed the logistic growth model [2] to overcome this limitation and this is still widely used to describe density dependent growth. Many other models have been proposed to describe population dynamics [3,4,5,6] but it is not clear which model is most appropriate in any given situation; the logistic model is the default choice in the absence of other data. Models of population dynamics are not merely interesting in isolation. For example in our field, epidemiology, adding birth and death of individuals to a model of infectious disease spread can alter the dynamics of the epidemic. Therefore, getting a suitable model of population growth is an important step in producing realistic models of disease spread which can be analysed to provide predictive information about potential impact of epidemics, and to evaluate control strategies.

Process algebra has increasingly been used to model a wide range of biological systems [7,8,9,10,11]. The benefits of using process algebras to study such systems are twofold. First, process algebra allows formal, precise and unambiguous

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 139–155, 2008.

expression of a model. Second, process algebra has a formal mathematical semantics, allowing rigorous investigation of the model via a range of techniques. For example, our work uses the discrete time process algebra *Weighted Synchronous Calculus of Communicating Systems* (WSCCS)[12]. The underlying semantics of WSCCS can be viewed as a Discrete Time Markov Chain (DTMC). Simulation can be used to explore the model. Steady state analysis can be carried out, and properties of the Markov Chain computed, e.g. probability of being in a particular state, or average number of occurrences of an action before a specific event occurs. Such investigation can be computationally expensive. Our previous work [13,14] has been to facilitate further symbolic analyses of the model by developing a rewriting-based method to derive Mean Field Equations (MFEs) from a description of a system in WSCCS. The MFEs describe the average behaviour of the system at the population level and are analogous to traditional Ordinary Differential Equation (ODE) models of biological systems. The MFEs provide an approximation of the system dynamics of the DTMC corresponding to the WSCCS description. The derived MFEs are amenable to analysis using established algebraic techniques developed by mathematical biologists for ODEs.

The key advantage of our approach is that biological observations of individuals can be exploited in making the (individual based) WSCCS model, and the MFEs are derived automatically and efficiently. The alternative approach, used by mathematical biologists for many years, is to simply write down the MFE or ODE description assuming that behaviour at population level is well understood. While this formulation of the equations is backed up by experience, there is no rigorous relation between the actions of individuals and the outcome at a population level. Matching with disease data provides validation of ODEs, but many plausible terms can match the same data. In both approaches, facts about individual behaviour are abstracted to obtain population level equations capturing only information about the number in each class of individual. The difference is that our approach makes assumptions about behaviour explicit and that the method of abstracting is rigorous.

In this paper we consider the problem of accurately representing population growth using process algebra. Others have investigated individual based models of population dynamics and related their behaviour to population level equations. Sumpter [10] developed a simple WSCCS model of population growth and derived MFEs for the model. Brännström and Sumpter [15] presented individual based (not process algebra) models of competition which could be used to derive several existing population models but notably not Verhulst's logistic equation. The work presented here improves on previous work by applying a rigorous method across a range of different models of population growth.

Outline of the Paper. Section 2 gives a brief description of the syntax and semantics of WSCCS used in our models, and an outline of the method for deriving MFEs. In Sect. 3 WSCCS models of population dynamics are presented, which include density dependent growth in a variety of formulations (in either births or deaths, and introduced implicitly by enriching the WSCCS language or explicitly by including agents representing resources for which the population competes).

The resultant changes in overall population dynamics are explored, comparing the derived MFEs to traditional population level equations for population dynamics. In Sect. 4 we add disease spread to our model of population dynamics. Our results are summarised in Sect. 5.

2 Background

2.1 WSCCS Syntax and Semantics

In WSCCS the basic components are *actions* and the *processes* (or *agents*) that carry out those actions. The actions are chosen by the modeller to represent activities in the system. For example, *infect*, *send*, *receive*, *throw dice*, and so on. Actions form an abelian group with identity $\surd$ and the inverse of action a being $\overline{a}$. Actions occur instantaneously and have no duration. There is no notion of time in WSCCS, but there is ordering of events. WSCCS is a probabilistic process algebra, meaning that the decision to move from one state to another can be a probabilistic one. The formal syntax and semantics of WSCCS is presented in Tofts [12]. The main details are repeated here for the convenience of the reader.

The possible WSCCS expressions are given by the following BNF grammar:

$$A ::= X \mid a{:}A \mid \Sigma\{w_i.A_i | i \in I\} \mid A \times B \mid A\lceil L \mid \Theta(A) \mid A[S] \mid X \stackrel{\text{def}}{=} A\ .$$

Here $X \in Var$, a set of process variables; $a \in Act$, an action group; $w_i \in \mathscr{W}$, a set of weights; S a set of renaming functions, $S : Act \to Act$ such that $S(\surd) = \surd$ and $\overline{S(a)} = S(\overline{a})$; action subsets $A \subseteq Act$ with $\surd \in A$; and arbitrary indexing sets I. The informal interpretation of the operators is as follows:

- 0 a process which cannot proceed, representing deadlock;
- X the process bound to the variable X;
- $a{:}A$ a process which can perform the action a becoming the process A;
- $\Sigma\{w_i.A_i | i \in I\}$ the weighted choice between processes A_i , the weight of A_i being w_i . Considering a large number of repeated experiments of this process, we expect to see A_i chosen with relative frequency $w_i/\Sigma_{i\in I} w_i$. Weights are generally positive natural numbers or reals, but may also incorporate the special weight ω which is greater than all natural numbers. This is used in *priority* and is written $m\omega^n$ where $m, n \geq 0$. The binary plus operator can be used in place of the indexed sum i.e. writing $\Sigma\{1_1.a{:}0, 2_2.b{:}0 | i \in \{1, 2\}\}$ as $1.a{:}0 + 2.b{:}0$;
- $A \times B$ the synchronous parallel composition of A and B. At each stage each process must perform an action with the composed process performing the composition (denoted #) of the individual actions, e.g. $a : A \times b : B$ yields $a\#b{:}(A \times B)$. This is a powerful operator: models are constructed by describing simple individuals and composing a number of those in parallel. McCaig [13] introduces an extended notation $A\{n\}$ which is syntactic sugar for n instances of process A in parallel, where $n \in \mathbb{N}$;

Table 1. Operational rules for WSCCS

$\frac{}{a{:}A \xrightarrow{a} A}$	$\frac{}{\sum\{w_i.A_i \mid i \in I\} \overset{w_i}{\longmapsto} A_i}$
$\frac{A \xrightarrow{a} A' \quad B \xrightarrow{b} B'}{A \times B \xrightarrow{a \# b} A' \times B'}$	$\frac{A \overset{w}{\longmapsto} A' \quad B \overset{v}{\longmapsto} B'}{A \times B \overset{wv}{\longmapsto} A' \times B'}$
$\frac{A \xrightarrow{a} A' \quad B \overset{w}{\longmapsto} B'}{A \times B \overset{w}{\longmapsto} A \times B'}$	$\frac{A \overset{w}{\longmapsto} A' \quad B \xrightarrow{a} B'}{A \times B \overset{w}{\longmapsto} A' \times B}$
$\frac{A \xrightarrow{a} A' \quad a \in L}{does_L(A)}$	$\frac{A \overset{w}{\longmapsto} A' \quad does_L(A')}{does_L(A)}$
$\frac{A \xrightarrow{a} A' \quad a \in L}{A \lceil L \xrightarrow{a} A' \lceil L}$	$\frac{A \overset{w}{\longmapsto} A' \quad does_L(A')}{A \lceil L \overset{w}{\longmapsto} A' \lceil L}$
$\frac{A \xrightarrow{a} A'}{A[S] \xrightarrow{S(a)} A'[S]}$	$\frac{A \overset{w}{\longmapsto} A'}{A[S] \overset{w}{\longmapsto} A'[S]}$
$\frac{A \xrightarrow{a} A' \quad X \overset{\text{def}}{=} A}{X \xrightarrow{a} A'}$	$\frac{A \overset{w}{\longmapsto} A' \quad X \overset{\text{def}}{=} A}{X \overset{w}{\longmapsto} A'}$
$\frac{A \xrightarrow{a} A'}{\Theta(A) \xrightarrow{a} \Theta(A')}$	$\frac{A \overset{n\omega^i}{\longmapsto} A' \nexists (j>i).A \overset{m\omega^j}{\longmapsto}}{\Theta(A) \overset{n}{\longmapsto} \Theta(A')}$

- $A \lceil L$ a process which can only perform actions in the group L. This operator is used to enforce communication on actions $b \notin L$. Two processes in parallel may communicate when one carries out an action and the other carries out the matching co-action, e.g. *infect* and $\overline{infect}$. Communication can be used to model passing of information from one process to another, or to coordinate activity. Such communication is strictly two-way; that is, only two processes may interact on this action;
- $\Theta(A)$ represents taking the prioritised parts of the process A only;
- $A[S]$ represents A relabelled by the function S (we do not use relabelling in this paper, but it is included for completeness);
- $X \overset{\text{def}}{=} A$ represents binding the process variable X to the expression A.

The semantics of WSCCS is transition based, defining the actions that a process can perform and the weight with which a state can be reached. The operational rules of WSCCS, presented in Table 1, formalise the descriptions above. In particular note the two different arrows which feature in the table: $\xrightarrow{a}$ represents a transition associated with the action a ; and $\overset{w}{\longmapsto}$ represents a transition associated with a weight w . The auxiliary predicate $does_L(A)$, which denotes the ability of A to perform L after zero or more probabilistic actions, is well defined since only finitely branching choice expressions are allowed.

2.2 Deriving Mean Field Equations from WSCCS Models

In McCaig's thesis [13] and the related report [14] a method is described to automatically derive Mean Field Equations from WSCCS models. We give an

$$N \stackrel{\text{def}}{=} p_d.\surd : 0 + p_b.\surd : (N \times N) + (1 - p_d - p_b).\surd : N$$
$$\mathit{Population} \stackrel{\text{def}}{=} N\{n\}\lceil\{\surd\}$$

Fig. 1. Naive population model

overview of the approach here to aid understanding of the following sections. Sample derivations are given at the end of this section and in Sect. 3.2.

Consider the simple model of population growth in Fig. 1. The N agents die with probability p_d, becoming the null agent 0, give birth with probability p_b, becoming the agent consisting of two N agents in parallel, or do neither with probability $(1 - p_d - p_b)$, remaining as a single agent N. The model can be simulated, producing a single trace through the dynamics of the system. A second simulation may of course produce quite different behaviour since this is a stochastic process; therefore, of more interest is the *average* behaviour of the system as time progresses. This can be obtained by averaging the time series results of repeated simulations of the system. Clearly this becomes time-consuming, as the number of processes n and number of repetitions increases. An alternative is to generate the whole transition system for the model and to average the states produced, but as n increases the state space grows exponentially.

McCaig's method avoids generating the state space of the whole system, instead applying transformations to the WSCCS expression of the model, yielding an approximation (average) of the transition system in the form of first-order mean field equations. The approximation is shown to be "good" (i.e. lies within the standard deviation when compared with repeated simulations) in McCaig's thesis. Further, when the system becomes infinitely large, the mean of the DTMC corresponding to the transition system is proved to be equivalent to the derived MFEs. Larger populations eliminate the stochastic effects associated with low copy numbers.

The advantages of our approach are: the computational expense of generating the state space and/or simulation is avoided (the method is $O(a^2c)$ where a is the number of agents and c is the number of actions in the WSCCS description); it is a symbolic approach (avoiding questions regarding the exploration of the parameter space); and the MFEs, being a different view of the system and amenable to further analysis, offer new insight to the system.

The method applies to models of the form $A1\{n_1\}|...|Am\{n_m\}$ where the Ai communicate with each other (usually on a subset of actions). Models are limited in that steps involving probabilistic choice between actions must be separate from steps involving communication (which must have branches weighted 1).

Independently, the PEPA group [16,17] and Cardelli [18] have proposed methods for deriving ODEs from process algebra. Their work differs in that their process algebras are continuous, based on rates rather than probabilities. Two of the methods are based on a mass action assumption, and not tied to the standard process algebra semantics. In contrast, our goal has been to preserve this

association, so that understanding individuals and their interactions translates automatically to population behaviour via process algebra semantics.

Transition Table: Relating Actions to Overall System Evolution. The transition system may be viewed as the evolution of the state vector $A1\{n_1\}|...|Am\{n_m\}$ For a particular Ai an action has three possible effects:

Exit activity: Following the action, the process evolves to some other agent Aj therefore the number of Ai agents is decreased.
Entry activity: In symmetry with an exit activity for Ai above must be an entry activity for Aj. The number of Aj agents increases.
None: The process becomes Ai and there is no change in number of Ai agents.

WSCCS is a synchronous calculus, therefore in each time step, for every agent in the system, one of the above activities will occur. Our method is based around construction and interpretation of a transition table TT noting these exit and entry activities (Fig. 2).

for each agent Ai {
  for each $(w_j.a_j : Ak) \in$ *transitions(Ai)* {
    for each $Am \in$ *components(Ak)*
      TT[(Ai, a_j*),Am] = TT[(Ai,* a_j*),Am] + calculateTerm(Ai,* w_j, a_j*)*
  } }

Fig. 2. Constructing the transition table from a WSCCS model

The rows of TT denote the agents Ai at time t and their enabled actions aj. The columns of the transition table denote the agents Ak at the next time $t+1$. The term in cell $(Ai\ aj, Ak)$ is the proportion of Ai_t agents performing aj to become Ak_{t+1}. The derivation of this term is fully determined (see description below) by the context of the action carried out (e.g. part of a probabilistic choice, or part of a communication) and the composition of the population (i.e. how many of each different agent there are). Where Ai evolves to the same agent Ak irrespective of which action it performs a single row is used for that agent which is labelled $Ai\ *$. An example is the $F1$ agent in Fig. 4. The mean field equation for Ak_{t+1} is obtained by summing the terms in the column Ak.

Some auxiliary definitions are required. Processes can be classified by syntactic features as: *communicating* (having an action enabled that is involved in a communication), *probabilistic* (having only actions enabled that are not involved in communication), and *priority* (communicating and using ω weights). Given a *serial* process $A = w_1.a_1 : A1 + w_2.a_2 : A2 + ... + w_n.a_n : An$ define $transitions(A) = \{w_1.a_1 : A1, w_2.a_2 : A2, ..., w_n.a_n : An\}$. Given a *parallel* process $A = A1 \times A2 \times ... \times An$ define $components(A) = \{A1, A2, ...An\}$. For a process communicating on action a, we define two groups of agents involved in the collaboration: *collaborators* are those processes with the matching action $\overline{a}$, and *competitiors* are those processes with the same action a.

```
function calculateTerm (A, w, a): String {
  case A in {
    probabilistic(A): return w * A_t;
    communicating(A) and priority(A):
      term = (A_t * collaborators(A))/(A_t + competitors(A));
      if a equals √ return (A - term) else return term;
    communicating(A) and not priority(A):
      term = (A_t * collaborators(A))/(A_t + collaborators(A) + competitors(A));
      if a equals √ return (A - term) else return term;
}}
```

Fig. 3. Pseudo code to calculate proportion of agents at time $t + 1$

The pseudo code to compute the terms in the table is given in Fig. 3. For probabilistic choice, the semantics of WSCCS (Table 1) specifies that over a number of experiments the different branches will be taken in numbers consistent with their weights. For convenience, the weights in such choices sum to 1 in the models in this paper hence the term is simply wA_t. For communication, McCaig enumerates the possible outcomes based on a classification of modes of communication (prioritised or not, single action a or multiple actions e.g. $a\#a\#a$). This results in complex formulae based on the weighted multinomial choice of those outcomes giving the average number of communications. For single actions, as used in this paper, these formulae can be simplified. These are the formulae used in the *calculateTerm* function of Fig. 3. The full version of the approach [13,14] assumes weights do not have to sum to 1, and also gives the formulae for multiple action communications.

Derivation of MFE for a Simple Population Growth Model. Consider again the simplistic model of population growth given in Fig. 1. The actions in Fig. 1 are simply $\surd$. That is, activities are of no interest, only the evolution of numbers of agents is significant. As in all of our models, the system as a whole is described by the system equation *Population*, comprising multiple copies of each kind of agent in parallel.

The transition table for this system is as follows:

	0	N_{t+1}
$(N_t, \surd)$	$p_d N_t$	$(1 - p_b - p_d)N_t + 2p_b N_t$

Each column leads to a MFE for that agent, but 0 is ignored here since this is not of interest to us. The method outlined above generates the following MFE

$$N_{t+1} = (1 + p_b - p_d)N_t \tag{1}$$

where N_{t+1} represents the number of N agents at time $t + 1$ expressed in terms of N_t, the number of N agents at time t. Since this model has no communication between agents, and a single step with probabilistic choice, the derived MFE can be easily verified manually.

3 Density Dependent Growth

Equation (1) describes a simple recurrence relation. With $p_b > p_d$ the population will become infinitely large; $p_b < p_d$ will lead to the population dying out, while $p_b = p_d$ will lead to an equilibrium state for any initial population size, $N_0 = n$. The probabilities p_b and p_d are fixed, therefore the average behaviour of this model is similar to that of the simple exponential growth model described by Malthus [1]. Biologically, it is more realistic to consider a model in which the probability of birth and death vary depending on the size of the population at each instant in time (density dependence). For example, as the population grows food and shelter become scarce, therefore individuals become weaker and are more likely to die. Alternatively this weakness may manifest itself as a reduced fecundity and a reduction in the birth rates. This section presents several ways of modelling these notions in WSCCS, obtaining more realistic models of population growth.

3.1 Functional Probabilities

What is required is the ability to modify p_b and/or p_d on the fly as the population changes. The first method described here is to add assumptions about how probability of birth and death depend on population size using *functional probabilities* [13]. Functional probabilities add considerable convenience and elegance of expression to complex models, while adding no new semantic concepts to WSCCS. Functional probabilities are implemented by encoding population size as part of the agent name, a technique [19] commonly used in process algebra. The size of the resultant model is much increased, and the translation itself is unremarkable: the interested reader is referred to [13] for the full details.

Instead of fixed probabilities, a functional definition is given. For example, probability p_x can be made a function f of the number of A agents (denoted $[A]$) by

$$p_x \stackrel{\text{def}}{=} min(max(0, f([A])), p_L).$$

The function may take any format required, since it appears directly in the MFEs and is often not computed numerically. The probability p_L is the upper limit for p_x, chosen to ensure that all probabilities in the model are always in the range $0 \leq p \leq 1$. The *min* and *max* expressions may be required to ensure that p_x is in the allowed range, but these terms make mathematical analysis of the MFEs more complex. Often, in our further analysis we assert $p_x = f([A])$ based on very low likelihood of reaching a state where *min* and *max* are not satisfied by f (therefore those states make little contribution to the *average* behaviour captured by the MFEs).

Density Dependent Birth. Density dependent birth can be added to the model in Fig. 1 by making the probability of birth p_b inversely proportional to $[N]$.

$$p_b \stackrel{\text{def}}{=} min(max(0, p_{b_0} - k * [N]), p_L),$$

where p_{b_0} is the probability of birth in the absence of crowding and k is a measure of the strength of the effect of crowding, $0 < k \ll 1$.

Using the method of Sect. 2.2, the MFE derived is

$$\begin{aligned} N_{t+1} &= N_t + (p_{b_0} - kN_t - p_d)N_t \\ &= N_t + (p_{b_0} - p_d)N_t\left(1 - \frac{kN_t}{p_{b_0} - p_d}\right). \end{aligned} \tag{2}$$

This is our first realistic model of population growth, derived from an expression of individual behaviour. Compare this to the discrete time version of Verhulst's logistic equation

$$N_{t+1} = N_t + rN_t\left(1 - \frac{N_t}{K}\right). \tag{3}$$

where r represents reproductive rate and K the carrying capacity of the population. Simple substitution of $r = (p_{b_0} - p_d)$ and $K = (p_{b_0} - p_d)/k$ in (3) yields (2). The logistic equation is the most commonly used equation for describing population dynamics and is frequently included as a self limiting growth term in models of disease spread. This gives confidence in our approach, and endorses Verhulst's equation.

Density Dependent Death. Density dependent death can similarly be added to Fig. 1 by choosing probability of death p_d directly proportional to $[N]$ with

$$p_d \stackrel{\text{def}}{=} min(max(0, p_{d_0} + k * [N]), p_L),$$

where p_{d_0} is the probability of death in the absence of crowding. The MFE, derived once again using our method,

$$\begin{aligned} N_{t+1} &= N_t + (p_b - (p_{d_0} + kN_t))N_t \\ &= N_t + (p_b - p_{d_0})N_t\Big(1 - \frac{kN_t}{p_b - p_{d_0}}\Big), \end{aligned}$$

is equivalent to the logistic equation with $r = (p_b - p_{d_0})$ and $K = (p_b - p_{d_0})/k$.

Summary. The results above are pleasing: we have shown that it is possible to derive the logistic equation from an individual based model of population growth. This contradicts the findings of Brännström and Sumpter [15] who did not find the logistic equation for any of their models. Our results should not be surprising: in the functional probabilities we are making the probabilities linearly proportional to the population size, effectively encoding the same assumptions which lead to the logistic equation in the traditional population level models. It would have been more surprising if we had not derived the logistic equation.

3.2 Food as an Explicit Resource

The advantage of individual based modelling techniques is that population level assumptions can be avoided, to be replaced by population level behaviours arising from the explicit individual interactions. To the models seen so far we add

$$
\begin{aligned}
N1 &\stackrel{\text{def}}{=} 1.eat : (N2 \times N2) + 1.\surd : N2 \\
F1 &\stackrel{\text{def}}{=} 1.\overline{eat} : F2 + 1.\surd : F2 \\
N2 &\stackrel{\text{def}}{=} p_d.\surd : 0 + (1 - p_d).\surd : N1 \\
F2 &\stackrel{\text{def}}{=} 1.\surd : F1 \\
\mathit{Population} &\stackrel{\text{def}}{=} N1\{n\} \times F1\{f\} \lceil \{\surd\}
\end{aligned}
$$

Fig. 4. Density dependence on births with non-prioritised communication

agents representing "food", i.e. some finite resource required by individuals to survive, and for which there is competition. Any other similar resource, e.g. space, can be modelled in exactly the same way. Access to this resource can be used to determine the likelihood of either birth or death.

Acquiring a resource is modelled in WSCCS by communication between food agents and individuals, requiring the use of more complex language features than seen in the models so far. Two forms of communication are available: prioritised and non-prioritised. Using prioritised communication between the food agents and the population agents forces individuals to eat; however, in a population it is likely that some individuals, while foraging, may fail to find food which is present. Using non-prioritised communication models the possibility that individuals fail to eat even when food is present and is therefore more biologically plausible. As above, models exploring density dependence on births and density dependence on deaths are considered separately.

Density Dependence on Births. The model given in Fig. 4 has individuals in the population competing for the available food resource (the *eat* action), giving birth after eating, and dying probabilistically.

The agents $N1$ and $N2$ represent the members of the population at the different stages of the model. The $N1$ agents can eat and become the parallel agent $N2 \times N2$, representing birth. If they do not eat the $N1$ agents become a single $N2$ agent. In the second stage of the model the $N2$ agents make a probabilistic choice to die or survive. The total number of food agents is constant therefore the F agents $(F1, F2)$ should be thought of as units of food which the environment can produce in a time step rather than discrete portions of food which are consumed by the population.

For such models, the method generates MFE for all agents, i.e. $N1, N2, F1$, $F2$, where $N1$ is expressed in terms of $N2$ and vice versa. Similarly for $F1$ and $F2$. Generally we are interested only in a complete cycle of behaviour. That is, starting with agents $N1$, evolving to agents $N2$, then back to $N1$ (two stages here). We take the $N1$ equation, substitute to remove occurrences of $N2$ and obtain an equation only in $N1$ (and $F1$). Finally, we rename $N1$ as simply N. The fact that the number of food agents remains constant means that the derived MFE for $F1$ can be simplified to f in the MFE for N.

Deriving the terms of the MFEs for this model is more complex: although the definition of $N1$ suggests the choice to *eat* or not is equally weighted, in fact this choice is also influenced by availability of $F1$ agents with which to synchronise. This is reflected in the *calculateTerm* function described in Sect. 2.2. For example, here it is possible that no individuals eat (with very low probability), or that all do (assuming $[N1] \leq [F1]$) (also with low probability), or all of the possibilities inbetween. As explained earlier, the *calculateTerm* function yields a formula based on the weighted multinomial choice of those possible outcomes. The method yields the following transition table. Note that the term for the communicating action (*eat*) reflects that $N1$ *collaborates* with $F1$ but has no *competitors* for the action.

	0	$N1_{t+1}$	$N2_{t+1}$	$F1_{t+1}$	$F2_{t+1}$
$(N1_t, eat)$			$2\frac{N1_t * F1_t}{N1_t + F1_t}$		
$(N1_t, \surd)$			$N1_t - \frac{N1_t * F1_t}{N1_t + F1_t}$		
$(F1_t, *)$					$F1_t$
$(N2_t, \surd)$	$p_d N2_t$	$(1-p_d)N2_t$			
$(F2_t, *)$				$F2_t$	

Summing the columns and simplifying as described above leads to the MFE

$$N_{t+1} = (1-p_d)N_t + \frac{(1-p_d)fN_t}{f+N_t}. \tag{4}$$

Here the term $(1-p_d)N_t$ represents the mean proportion of the existing population which survives the probabilistic death stage. The term $fN_t/(f+N_t)$ represents the mean number of new births with the factor $(1-p_d)$ representing the proportion of new births which survive the probabilistic death stage. We find the steady state of this model by setting $N_{t+1} = N_t = N^*$:

$$N^* = (1-p_d)N^* + \frac{(1-p_d)fN^*}{f+N^*}.$$

Solving for N^* we get

$$N^* = \frac{(1-2p_d)f}{p_d}.$$

For biological realism the steady state should be positive, therefore $p_d < 0.5$. Note that this fact is not obvious from the WSCCS model, but becomes clear in the MFE. The values of these probabilities can be observed in the field, but an important factor is the length of timestep. If we need to reduce p_d to meet the above requirement we can reduce the timestep represented by our models and adjust all other parameters accordingly.

Sumpter [10] developed a mechanism for describing self limiting growth in a population which made use of food as an agent. He derived the following MFE using an heuristic

$$N_{t+1} = (1-p_d)N_t + \min[(1-p_d)N_t, f],$$

$$N1 \stackrel{\text{def}}{=} 1.eat : N2 + 1.\surd : 0$$
$$F1 \stackrel{\text{def}}{=} 1.\overline{eat} : F2 + 1.\surd : F2$$
$$N2 \stackrel{\text{def}}{=} p_b.\surd : (N1 \times N1) + p_d.\surd : 0 + (1 - p_b - p_d).\surd : N1$$
$$F2 \stackrel{\text{def}}{=} 1.\surd : F1$$
$$Population \stackrel{\text{def}}{=} (N1\{n\} \times F1\{f\})\lceil\{\surd\}$$

Fig. 5. Density dependence on deaths with non-prioritised communication

where p_d is the probability of death in any timestep and f is the number of food agents. The underlying assumptions of this model are undesirable biologically: individuals are guaranteed to find food if it is available because prioritised communication is used. Therefore, every member of the population will give birth at each step of time until the size of the population is larger than the number of food agents, after which the number of births will be equal to the number of food agents. This model has a stable steady state of $N^* = f/p_d$, when $p_d \leq 0.5$, which is larger than for our model.

Density Dependence on Deaths. In Fig. 5 the $N1$ agents can once again eat, becoming the agent $N2$, but here if they do not eat they die, becoming the null agent 0. The $N2$ agents then give birth probabilistically and, to be realistic, can also die probabilistically. That is, in each step of time a proportion of the population die, for instance, due to age and some die due to a lack of food. The MFE for this model is

$$N_{t+1} = (1 + p_b - p_d)\frac{fN_t}{f + N_t}, \tag{5}$$

where term $fN_t/(f + N_t)$ represents the proportion of the population which eat and therefore survive the competition for food, with the factor $(1 + p_b - p_d)$ representing the increase in the population due to births and the decrease due to probabilistic death. Equation (5) can be rearranged to give

$$N_{t+1} = \frac{aN_t}{1 + bN_t}, \tag{6}$$

where $a = (1 + p_b - p_d)$ and $b = 1/f$. Equation 6 is the Beverton-Holt model [3], originally proposed as a model of salmon populations displaying density dependent birth; however, we have derived this equation from an individual based model featuring density dependent death. Our derivation endorses the plausibility of the Beverton-Holt model, which is commonly used in models of plant populations but not so widely used for animal populations.

Setting $N_{t+1} = N_t = N^*$ in (5) and solving for N^* yields the steady state

$$N^* = (p_b - p_d)f.$$

In this case to ensure the steady state is positive we require $p_b > p_d$.

$$
\begin{aligned}
N1 &\stackrel{\text{def}}{=} p_b.\surd : (N2 \times B2) + p_d.\surd : D2 + (1 - p_b - p_d).\surd : N2 \\
F1 &\stackrel{\text{def}}{=} 1.\surd : F2 \\
N2 &\stackrel{\text{def}}{=} 1.eat : N1 + 1.\surd : 0 \\
F2 &\stackrel{\text{def}}{=} 1.\overline{eat} : F1 + 1.\surd : F1 \\
B2 &\stackrel{\text{def}}{=} 1.\surd : N1 \\
D2 &\stackrel{\text{def}}{=} 1.eat : 0 + 1.\surd : 0 \\
Population &\stackrel{\text{def}}{=} (N1\{n\} \times F1\{f\})\lceil\{\surd\}
\end{aligned}
$$

Fig. 6. Density dependence on deaths, with choice followed by communication

Order Matters? Clearly, changing the WSCCS model can affect the MFEs derived, but even a relatively small, intuitively negligible, change can make a difference. In the models considered in Figs. 4 and 5 the focus is on the two-stage behaviour of the $N1$ agents. This means that the communicative (eating) step is followed by the probabilistic step with births and deaths. We may naively assume that considering the two-stage behaviour of the $N2$ agents, thus reversing the order of the communicative and probabilistic steps, would lead to the same overall long term behaviour of the model. However, the derived MFE for the behavior of the $N2$ agents in Fig. 5 is

$$N_{t+1} = (1 + p_b - p_d)\frac{fN_t}{f + (1 + p_b - p_d)N_t},$$

where the denominator features a factor of $(1 + p_b - p_d)$ not present in (5).

This difference arises because changing the order in which the steps occur also changes the underlying biological assumptions of the model. The newborn individuals are now available to compete for the available food (leading to the $+p_b$ term) and the individuals which probabilistically die are not (leading to the $-p_d$ term). Generating a WSCCS model in which probabilistic choice is followed by a communicative phase is more complex than simply swapping these steps. A suitable model, which will lead to the MFE (5), can be seen in Fig. 6.

In Fig. 6 the agents which make the probabilistic choice to die enter a dying state, $D2$, where they compete for food and are then removed from the system, irrespective of whether they eat or not. The newly born individuals are in the state $B2$ which does not compete for food and becomes $N1$ at the next stage. This means that the overall mean two-stage behaviour of the $N1$ agents in Fig. 6 is the same as for the $N1$ agents in Fig. 5.

This simple example illustrates the importance of thinking carefully about the biological interpretation of actions in the WSCCS model, highlighted by the derivation of MFEs. This is particularly important when considering more complex models such as that in Sect. 4 which adds population dynamics to a model of infectious disease.

4 Population Dynamics and Disease

While population dynamics are interesting in their own right they are also crucial in developing realistic models of disease spread. The model in Fig. 7 adds infectious disease spread, based on the models of Norman and Shankland [8], to the density dependent death population dynamics of Fig. 5. In a typical disease model the population is divided into 3 groups: susceptibles (S) have never had the disease, infecteds (I) currently have the disease, and recovereds (R) have previously had the disease and are immune to future infection.

The first stage in the model is the eating stage in which $S0, I0$ and $R0$ all compete for food. Those that do not eat will die. The second stage is a contact stage in which infected (*Trans*) agents come into contact with the population and potentially pass the disease to susceptibles. The infected individuals are represented by parallel agents with the *Trans* agents passing on the disease and the $T1$ agents able to be contacted by a *Trans* agent. Communication is prioritised so that all *Trans* make contact. Prioritised contact is plausible biologically since contact with the whole population is possible (not just the susceptibles) and contact is not guaranteed to result in infection (see $SI2$). $S1$ that are contacted become $SI2$, while $T1$ and $R1$ agents are not affected by contact since infected and recovered individuals cannot become infected again. After the contact stage the *Trans* agents all become the null agent 0 so that the infected individuals are once again represented by a single agent. The final stage is the probabilistic stage in which all individuals can give birth to a susceptible individual, with probability p_b, or die, with probability p_d. In addition the $SI2$ agents become infected with probability p_a and $I2$ agents can recover with probability p_r.

$$
\begin{aligned}
S0 &\stackrel{\text{def}}{=} 1.\mathit{eat} : S1 + 1.\surd : 0 & \qquad S1 &\stackrel{\text{def}}{=} \omega.\mathit{infect} : SI2 + 1.\surd : S2\\
R0 &\stackrel{\text{def}}{=} 1.\mathit{eat} : R1 + 1.\surd : 0 & \qquad R1 &\stackrel{\text{def}}{=} \omega.\mathit{infect} : R2 + 1.\surd : R2\\
I0 &\stackrel{\text{def}}{=} 1.\mathit{eat} : (T1 \times \mathit{Trans}) + 1.\surd : 0 & \qquad T1 &\stackrel{\text{def}}{=} \omega.\mathit{infect} : I2 + 1.\surd : I2\\
\mathit{Food}0 &\stackrel{\text{def}}{=} 1.\overline{\mathit{eat}} : \mathit{Food}1 + 1.\surd : \mathit{Food}1 & \qquad \mathit{Trans} &\stackrel{\text{def}}{=} \omega.\overline{\mathit{infect}} : 0 + 1.\surd : 0\\
\mathit{Food}1 &\stackrel{\text{def}}{=} 1.\surd : \mathit{Food}2\\
\mathit{Food}2 &\stackrel{\text{def}}{=} 1.\surd : \mathit{Food}0\\
S2 &\stackrel{\text{def}}{=} p_b.\surd : (S0 \times S0) + (1 - p_b - p_d).\surd : S0 + p_d.\surd : 0\\
SI2 &\stackrel{\text{def}}{=} p_b.\surd : (S0 \times S0) + p_a.\surd : I0 + (1 - p_a - p_b - p_d).\surd : S0 + p_d.\surd : 0\\
I2 &\stackrel{\text{def}}{=} p_b.\surd : (I0 \times S0) + p_r.\surd : R0 + (1 - p_r - p_b - p_d).\surd : I0 + p_d.\surd : 0\\
R2 &\stackrel{\text{def}}{=} p_b.\surd : (R0 \times S0) + (1 - p_b - p_d).\surd : R0 + p_d.\surd : 0\\
\mathit{Population} &\stackrel{\text{def}}{=} \Theta((S0\{s\} \times I0\{i\} \times \mathit{Food}0\{f\})\lceil\{\surd\})
\end{aligned}
$$

Fig. 7. SIR model with density dependence on deaths

The system of MFEs derived from this model is

$$\begin{aligned} S_{t+1} &= \frac{f}{f+N_t}\Big((1-p_d)S_t + p_b N_t - \frac{p_a S_t I_t}{N_t}\Big) \\ I_{t+1} &= \frac{f}{f+N_t}\Big((1-p_d-p_r)I_t + \frac{p_a S_t I_t}{N_t}\Big) \\ R_{t+1} &= \frac{f}{f+N_t}\Big((1-p_d)R_t + p_r I_t\Big), \end{aligned} \tag{7}$$

where $N_t = S_t + I_t + R_t$, the total population size at time t. These are similar to the standard SIR equations with frequency dependent transmission of disease [20], a form arising naturally from WSCCS models [8]. Here, however, there is an extra factor of $f/(f+N_t)$ on each equation that is the proportion of the population successfully eating. This is unconventional since in traditional models the transmission term (in this case $(p_a S_t I_t)/N_t$) is not affected by the density dependent birth or death term. We emphasise that the population dynamics of (7) come directly from explicit representation of individuals competing for food rather than any population level assumptions imposed on the model. These equations are therefore candidates for modelling population dynamics in disease systems, despite the differences to traditional models.

In contrast, if we had taken the population dynamics from Sect. 3.1, with functional probability of birth, and added disease as above, we would merely add a logistic term to the equation for S with each group also dying probabilistically. This result would be closer to the traditional ODE models. The advantage of this approach is that the nonlinear density dependent term only appears in one equation (S), therefore the equations are simpler and easier to analyse mathematically than (7) which contains nonlinear terms in all equations. The disadvantage of basing a disease model on the functional probability models of population growth is that the latter are based on assumptions about population growth which may be incorrect.

5 Conclusion

In this paper we have presented population dynamics models in which the population will, over time, tend to some steady state and will not display unbounded growth. There are two distinct types of model: those in which the effects of restricted resources are implicitly included by allowing more complex language features in the model (functional probabilities) and those in which those resources are explicitly represented by agents. The introduction of functional probabilities allow us to succinctly take full advantage of the expressive capabilities of WSCCS. These models led naturally to the logistic equation [2], the classical expression used to describe population dynamics. This is in contrast to the results of Brännström and Sumpter [15] who found several other existing expressions could be derived from their individual based models but not the logistic equation. The logistic equation arises from our models because the assumptions used

to introduce density dependence – functional probabilities which are linearly proportional to the population size – match the assumptions on which the logistic equation is based. If we use functional probabilities which are non-linearly proportional to the population size we would of course obtain different MFEs. It can be easily argued that adding functional rates is self-defeating for our objectives; if we allow inclusion of strong implicit assumptions, such as the nature of population growth, then we may as well simply write down the MFEs directly.

In order to reduce the number of population level assumptions in our models we have also developed models which feature agents to represent food, with the dynamics in the population arising from the competition between individuals for food. With density dependent death this model leads to the Beverton-Holt model [3] which was proposed for the population dynamics of fish stocks. The fact that this equation has naturally arisen here from the competition between individuals means we can consider the Beverton-Holt model a serious candidate to be used when modelling population dynamics. Further investigation including matching with data is required.

Lastly, our goal in population modelling is to incorporate models of disease to gain a more realistic individual based disease model. By adding a model of disease spread to population dynamics we have derived a system of equations (7) which differs from those which have previously been described in the literature. Because the population dynamics in our model naturally arise from the interactions between individuals and the environment, rather than any assumptions we have imposed on the population dynamics, we have well-founded reason to propose this model for a disease system featuring density dependence in deaths. As above, future work will include validating our models with disease data.

Acknowledgements. This work was supported by EPSRC through a Doctoral Training Grant (CM, from 2004–2007), and through *System Dynamics from Individual Interactions: A process algebra approach to epidemiology* (EP/E006280/1, all authors, 2007–2010).

References

1. Malthus, T.: An essay on the principle of population (1798)
2. Verhulst, P.: Notice sur la loi que la population suit dans son accroissement. Corr. Math. et Phys. 10, 113–121 (1838)
3. Beverton, R., Holt, S.: On the dynamics of exploited fish populations. Fisheries Investigations, Series 2. H.M.S.O., vol. 19 (1957)
4. Gompertz, B.: On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. Philosophical transactions of the Royal Society of London Series B 115, 513–585 (1825)
5. Hassell, M.: Density-dependence in single-species populations. Journal of Animal Ecology 45, 283–296 (1975)
6. Ricker, W.: Stock and recruitement. Journal of the Fisheries Research Board of Canada 11, 559–623 (1954)

7. Calder, M., Gilmore, S., Hillston, J.: Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. In: Proceedings of BioCONCUR 2004. Electronic Notes in Theoretical Computer Science. Elsevier, Amsterdam (2004)
8. Norman, R., Shankland, C.: Developing the use of process algebra in the derivation and analysis of mathematical models of infectious disease. In: Moreno-Díaz Jr., R., Pichler, F. (eds.) EUROCAST 2003. LNCS, vol. 2809, pp. 404–414. Springer, Heidelberg (2003)
9. Regev, A., Panina, E., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: an abstraction for biological compartments. Theoretical Computer Science 325, 141–167 (2004)
10. Sumpter, D.: From Bee to Society: an agent based investigation of honeybee colonies. PhD thesis, UMIST (2000)
11. Tofts, C.: Using process algebra to describe social insect behaviour. Transactions of the Society for Computer Simulation 9, 227–283 (1993)
12. Tofts, C.: Processes with probabilities, priority and time. Formal Aspects of Computing 6, 536–564 (1994)
13. McCaig, C.: From individuals to populations: changing scale in process algebra models of biological systems. PhD thesis, University of Stirling (2008), `www.cs.stir.ac.uk/~cmc/thesis.ps`
14. McCaig, C., Norman, R., Shankland, C.: Deriving mean field equations from process algebra models. Technical Report 175, University of Stirling (2008), `www.cs.stir.ac.uk/research/publications/techreps`
15. Brännström, A., Sumpter, D.: The role of competition and clustering in population dynamics. Proceedings of the Royal Society of London Series B 272, 2065–2072 (2005)
16. Calder, M., Gilmore, S., Hillston, J.: Automatically deriving ODEs from process algebra models of signalling pathways. In: Proceedings of CMSB 2005 (Computational Methods in Systems Biology), pp. 204–215 (2005)
17. Hillston, J.: Fluid Flow Approximation of PEPA models. In: QEST 2005, Proceedings of the 2nd International Conference on Quantitative Evaluation of Systems, pp. 33–42. IEEE Computer Society Press, Torino (2005)
18. Cardelli, L.: On process rate semantics. Theoretical Computer Science 391, 190–215 (2008)
19. Milner, R.: Communication and Concurrency. Prentice Hall, Englewood Cliffs (1989)
20. Begon, M., Bennet, M., Bowers, R., French, N., Hazel, S., Turner, J.: A clarification of transmission terms in host-microparasite models: numbers, densities and areas. Epidemiology and infection 129, 147–153 (2002)

Algebraic Analysis of Bifurcation and Limit Cycles for Biological Systems*

Wei Niu[1] and Dongming Wang[1,2]

[1] Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie – CNRS, 104 avenue du Président Kennedy, F-75016 Paris, France
[2] LMIB – SKLSDE – School of Science, Beihang University, Beijing 100083, China
wei.niu@etu.upmc.fr, Dongming.Wang@lip6.fr

Abstract. In this paper, we show how to analyze bifurcation and limit cycles for biological systems by using an algebraic approach based on triangular decomposition, Gröbner bases, discriminant varieties, real solution classification, and quantifier elimination by partial CAD. The analysis of bifurcation and limit cycles for a concrete two-dimensional system, the self-assembling micelle system with chemical sinks, is presented in detail. It is proved that this system may have a focus of order 3, from which three limit cycles can be constructed by small perturbation. The applicability of our approach is further illustrated by the construction of limit cycles for a two-dimensional Kolmogorov prey-predator system and a three-dimensional Lotka–Volterra system.

1 Introduction

Many physical and biological phenomena may be modeled mathematically by dynamical systems. For most nonlinear dynamical systems, it is difficult to find their exact analytical solutions (if such solutions exist at all). Therefore, to understand the phenomenon described by a complex dynamical system, it is necessary to study its behaviors such as stability, bifurcation, and limit cycles qualitatively. The qualitative analysis of such behaviors is a highly nontrivial task and for biological systems it is often performed experimentally by means of numerical simulation and visualization (see, e.g., [2]). Dynamical systems usually involve parameters and their behaviors may change dramatically as the parameters vary. Symbolic and algebraic computation provides powerful tools for rigorous qualitative analysis of dynamical systems with parameters.

We are concerned with biological phenomena that may be modeled by autonomous systems of ordinary differential equations of the form

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = p_1(\boldsymbol{\lambda}, x_1, \ldots, x_n), \ \ldots\ldots, \ \frac{\mathrm{d}x_n}{\mathrm{d}t} = p_n(\boldsymbol{\lambda}, x_1, \ldots, x_n), \tag{1.1}$$

where $p_1, \ldots, p_n$ are polynomials (or rational functions) in $\boldsymbol{\lambda}$ and $x_1, \ldots, x_n$ with real coefficients and $\boldsymbol{\lambda} = \lambda_1, \ldots, \lambda_m$ are real parameters independent of the derivation variable t. As usual, each x_i is a function of t.

* This work has been supported by the Chinese National Key Basic Research (973) Projects 2004CB318000 and 2005CB321901/2 and the SKLSDE Project 07-003.

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 156–171, 2008.

For biological systems of the form (1.1) with $p_1, \ldots, p_n$ being rational functions, it has been demonstrated in [15,16,23,24] how their steady states can be detected and how the stability of the steady states can be analyzed by using symbolic and algebraic computation. The technique of linearization used for stability analysis there may fail at bifurcation points because near such points the behavior of system (1.1) may differ qualitatively from that of its linearized system and bifurcation may occur (see [10,14,29]). Even for autonomous systems there may be many different bifurcating situations, whose study requires sophisticated mathematical methods and effective computational tools. Among them there is one important situation, called *Hopf bifurcation* (or *Andronov–Hopf bifurcation*). In this situation, the Jacobian matrix of system (1.1) has a pair of purely imaginary eigenvalues but no other eigenvalue with zero real part. Hopf bifurcation leads to the birth of limit cycles from a steady state of the dynamical system, when the steady state changes its stability via the movements of the imaginary eigenvalues away from the imaginary axis.

The behavior of limit cycles (i.e., isolated periodic orbits) has been observed in many physical and biological systems. The study of limit cycles is a subject of active research and of great interest in pure mathematics for more than a century. Determining the existence and the exact number of limit cycles is a difficult problem even for planar autonomous polynomial differential systems. This problem is closely related to the longstanding 16th problem of D. Hilbert [8,18,20,28,29]. The analysis of bifurcation and limit cycles is not only a challenging problem in the qualitative theory of dynamical systems, but also of practical value now for our understanding of the qualitative behaviors of biological systems.

In this paper, we analyze bifurcation and construct limit cycles for a concrete two-dimensional (planar) system, the self-assembling micelle system with chemical sinks [2], by using an algebraic approach based on triangular decomposition [21,25], Gröbner bases [3,9], discriminant varieties [11], real solution classification [27], and quantifier elimination by partial CAD [6]. The stability and bifurcation for this biological system have been analyzed initially in [16], where our general algebraic approach is described in generality. Here we show that this system may have a focus of order 3, from which three limit cycles can be constructed by small perturbation. The applicability of our approach is further illustrated by the construction of limit cycles for a Kolmogorov prey-predator system and a Lotka–Volterra system. The paper is structured as follows. In the following section, we explain how bifurcation and limit cycles for planar autonomous systems may be analyzed by using algebraic methods. The analysis of bifurcation and the construction of limit cycles for the self-assembling micelle system with chemical sinks are presented in detail in Section 3. Section 4 contains the results of analysis of bifurcation and limit cycles for the two-dimensional cubic Kolmogorov prey-predator system and the competitive three-dimensional Lotka–Volterra system. The paper concludes with a few remarks in Section 5.

Prior to the algebraic analysis of bifurcation investigated by the authors [16], other relevant work has been done by El Kahoui and Weber [7] who applied quantifier elimination [6] to Hopf bifurcation analysis, by Lu and others in [12,13]

where limit cycles for a class of biological systems are analyzed by using similar algebraic approaches, and by other researchers in the purely mathematical context of bifurcation and limit cycles (see, e.g., [8,18,19]). Hopf bifurcation analyzed in the mathematical context is usually for systems in the canonical form with the origin as their steady state. The biological systems with parameters that we need to analyze are not given in such a form. Transformation of the systems to the canonical form results in complicated expressions and the introduction of new variables and thus increases the computational complexity for the subsequent algebraic analysis considerably.

2 Bifurcation Analysis for Two-Dimensional Systems

In this section, we consider the case of Hopf bifurcation for $n = 2$ in which limit cycles may bifurcate. In this case, the characteristic polynomial of the Jacobian matrix of system (1.1) at a steady state has a pair of purely imaginary roots and the differential system is known as of *center-focus type*.

Let $n = 2$, the variables x_1 and x_2 be renamed x and y, $(\bar{x}, \bar{y})$ be a steady state of system (1.1), and $\bar{\boldsymbol{J}}$ be the Jacobian matrix

$$\begin{bmatrix} a(\boldsymbol{\lambda}, x, y) & b(\boldsymbol{\lambda}, x, y) \\ c(\boldsymbol{\lambda}, x, y) & d(\boldsymbol{\lambda}, x, y) \end{bmatrix}$$

of (1.1) at $(\bar{x}, \bar{y})$. Then the characteristic polynomial of $\bar{\boldsymbol{J}}$ has a pair of purely imaginary roots only if $(\bar{x}, \bar{y})$ satisfies the conditions

$$a + d = 0, \quad -a^2 - bc > 0. \tag{2.1}$$

The problems of deciding whether the steady states of system (1.1) without parameters satisfy the conditions (2.1) and determining the conditions on the parameters $\boldsymbol{\lambda}$ (when they are present) for the steady states of (1.1) to satisfy (2.1) may be solved by using algebraic methods based on quantifier elimination [6], real solution classification [27], and discriminant varieties [11].

Now assume that the conditions (2.1) are satisfied. We want to analyze the bifurcation and limit cycles for system (1.1) with $n = 2$. In the literature there are several methods for studying this problem [8], e.g., by using Poincaré–Birkhott normal forms, Liapunov constants, succession functions, averaging, and intrinsic harmonic balancing. To deal with the problem, we make a linear transformation

$$x = -\frac{1}{c}Y - \frac{a}{c\delta}X + \bar{x}, \quad y = -\frac{1}{\delta}X + \bar{y}, \quad t = \frac{\tau}{\delta}, \tag{2.2}$$

where $\delta = \sqrt{-a^2 - bc}$. Then system (1.1) with $n = 2$ is transformed into the following canonical form

$$\frac{\mathrm{d}X}{\mathrm{d}\tau} = Y + P(\boldsymbol{\lambda}, \delta, \bar{x}, \bar{y}, X, Y), \quad \frac{\mathrm{d}Y}{\mathrm{d}\tau} = -X + Q(\boldsymbol{\lambda}, \delta, \bar{x}, \bar{y}, X, Y), \tag{2.3}$$

where P and Q are polynomials in $\mathbb{Q}(\boldsymbol{\lambda}, \delta)[\bar{x}, \bar{y}, X, Y]$ and $\mathbb{Q}$ denotes the field of rational numbers.

According to Liapunov's theorem [14,29], if one can construct an analytic function $L(X,Y) \in \mathbb{Q}(\boldsymbol{\lambda},\delta)[\bar{x},\bar{y},X,Y]$, called a *Liapunov function*, such that $L(X,Y)$ is positive in the neighborhood of a steady state and the differential of $L(X,Y)$ along the integral curve of the differential system is definite, then the stability of the steady state can be determined by the sign of the differential. In [20], Wang described a procedure based on the classical method of J. H. Poincaré for constructing such a function for (2.3). Using this procedure, one can compute so-called *Liapunov constants* (or *focal values*) $v_3, v_5, \ldots, v_{2j+1}, \ldots$ in $\boldsymbol{\lambda}, \delta, \bar{x}, \bar{y}$ such that the differential $L(X,Y)$ along the integral curve of (2.3) is of the form

$$\frac{\mathrm{d}L(X,Y)}{\mathrm{d}\tau} = v_3 Y^4 + v_5 Y^6 + \cdots + v_{2j+1} Y^{2j+2} + \cdots .$$

Then the stability of the steady state $(0,0)$ for system (2.3) and thus $(\bar{x},\bar{y})$ for system (1.1) is determined by the sign of $\mathrm{d}L/\mathrm{d}\tau$ and therefore by the sign of the first nonzero Liapunov constant v_{2k+1}. Namely, we have the following simple criteria.

Theorem 1 ([20,29]). *For any given parametric values $\bar{\boldsymbol{\lambda}}$ of $\boldsymbol{\lambda}$ and steady state $(\bar{x},\bar{y}) = (\bar{x}_1,\bar{x}_2)$ of system (1.1) with $n = 2$,*

(a) *if there is an integer $k \geq 1$ such that $v_3 = \cdots = v_{2k-1} = 0$ but $v_{2k+1} \neq 0$, then $(\bar{x},\bar{y})$ is unstable when $v_{2k+1} > 0$, and asymptotically stable when $v_{2k+1} < 0$;*
(b) *if $v_{2j+1} = 0$ for all $j = 1, 2, \ldots$, then $(\bar{x},\bar{y})$ is stable of center type, but not asymptotically stable.*

In case (a), the steady state $(\bar{x},\bar{y})$ of system (1.1) is said to be a *focus of order k*. In case (b), $(\bar{x},\bar{y})$ is said to be a *center* of (1.1). By Theorem 1 (a), the problem of determining the stability of a focus is reduced to that of determining the signs of the Liapunov constants and thus may be solved by using the algebraic methods mentioned above.

It is possible to construct limit cycles in the neighborhood of a focus (so-called *small-amplitude limit cycles*) by perturbation. According to the fundamental theorem on bifurcation and limit cycles [1, p. 239], if the steady state $(0,0)$ of (2.3) is a focus of order k, then by perturbing (2.3), k small-amplitude limit cycles near $(0,0)$ may bifurcate. To construct such limit cycles, one needs to find the real solutions of $v_3 = \cdots = v_{2k-1} = 0, v_{2k+1} \neq 0$. This may be done by a combination of the methods and software tools of triangular decomposition, Gröbner bases, and real solving. When a real solution is found, one may perturb the system so that the Liapunov constants of the perturbed system have alternate signs, i.e., $v_i v_{i+2} < 0$ for $i = 3, \ldots, 2k+1$ in the neighborhood of $(0,0)$. Then k (and at most k) small-amplitude limit cycles may be created from the focus.

Determining necessary and sufficient conditions for $(0,0)$ to be a center from the computed Liapunov constants is a tougher issue because the conditions in Theorem 1 (b) are given by infinitely many equalities (in a finite number of variables). It may be tackled by using algebraic computation (as shown in the extensive literature on the derivation of center conditions, see, e.g., [22, Sect. 10.3])

together with some sophisticated mathematical techniques. The main difficulty in deriving center conditions and searching for differential systems having foci of high order from Liapunov constants comes from the large polynomials that cannot be effectively managed even on a powerful computer.

3 Bifurcation and Limit Cycles for Self-assembling Micelle Systems with Chemical Sinks

Consider the following dissipative dynamical system studied in [2,16]:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \mu - xy^2 - x(r+\alpha) = p, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = rx + xy^2 - \eta y = q. \tag{3.1}$$

The rate coefficients α and η represent combined quantities that include a common flow-rate component as well as separate chemical sink-rates for each species, and μ and r are intrinsic parameters.

We want to derive conditions for system (3.1) to be of center-focus type and for its foci to be stable and to construct small-amplitude limit cycles by means of bifurcation analysis.

3.1 Bifurcation Analysis

To determine the conditions under which system (3.1) is of center-focus, we first compute the purely lexicographical (plex) Gröbner basis of $\{p, q\}$ with $y \prec x$: the basis consists of two polynomials

$$g_1 = \eta y^3 - \mu y^2 + r\eta y + \alpha\eta y - r\mu, \quad g_2 = \alpha x + \eta y - \mu.$$

The system $g_1 = 0$, $g_2 = 0$ has real solutions for any parametric values of μ, r, and $\alpha \neq 0, \eta \neq 0$. Therefore, for $\alpha\eta \neq 0$ system (3.1) always has steady states. Let $\alpha\eta \neq 0$ and $\bar{y} = w$ be a real root of g_1. Then $\bar{x} = (\mu - \eta w)/\alpha$ is a real root of g_2. The Jacobian matrix of (3.1) at $(\bar{x}, \bar{y})$ is

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} -(w^2 + r + \alpha) & -\frac{2\,w(\mu-\eta w)}{\alpha} \\ r + w^2 & \frac{2\,w(\mu-\eta w)}{\alpha} - \eta \end{bmatrix}.$$

System (3.1) becomes of center-focus type when

$$f_1 = \alpha w^2 + 2\,\eta w^2 - 2\,\mu w + r\alpha + \alpha\eta + \alpha^2 = -(a+d)\alpha = 0,$$
$$f = (\alpha - \eta)w^2 - r\eta + r\alpha + \alpha^2 = a^2 + bc + (a+d)(r + w^2) < 0.$$

Note that

$$f_2 = g_1|_{y=w} = \eta w^3 - \mu w^2 + r\eta w + \alpha\eta w - r\mu = 0.$$

From $f_1 = f_2 = 0$, $f < 0$ and by using the Maple package DISCOVERER (developed by B. Xia, see http://www.is.pku.edu.cn/~xbc/software.html) or DV (developed by G. Moroz and F. Rouillier, see http://fgbrs.lip6.fr/Software/DV/), one can obtain conditions (CF) in the parameters η, μ, r, α for (3.1) to be of center-focus type, under which limit cycles may bifurcate from $(\bar{x}, \bar{y})$. The

conditions (CF) are quite complicated and we do not produce them here. It may also be proved easily (e.g., by using DISCOVERER) that, if $\eta = \alpha$, then there are no real values of $\mu, r, \alpha \neq 0$ that satisfy (CF). This confirms the conclusion in [2] that Hopf bifurcations are absent and (CF) hold only for non-physical values of α in this case. However, there do exist real values of μ, r, η, α such that $0 \neq \eta \neq \alpha \neq 0$ and (CF) hold, as we will see clearly below.

Under the conditions (CF), system (3.1) may be transformed by (2.2) into the following canonical form

$$\frac{\mathrm{d}X}{\mathrm{d}\tau} = Y + \frac{\delta}{\alpha}Q, \quad \frac{\mathrm{d}Y}{\mathrm{d}\tau} = -X + Q, \tag{3.2}$$

corresponding to (2.3), where $\delta = \sqrt{-f}$ and

$$\begin{aligned} Q &= \frac{\gamma}{(r+w^2)\delta^3}X^2 - \frac{2\,\alpha w}{(r+w^2)\delta^2}XY - \frac{\alpha(r+\alpha+w^2)}{(r+w^2)\delta^4}X^3 + \frac{\alpha}{(r+w^2)\delta^3}X^2Y, \\ \gamma &= 2\,\alpha w^3 + \eta w^3 - \mu w^2 + 2\,\alpha^2 w + 2\,r\alpha w + r\eta w - r\mu. \end{aligned} \tag{3.3}$$

The Liapunov constants of (3.2) may be computed by the function `miscel[licon]` from the Epsilon library [22] (see http://www-calfor.lip6.fr/~wang/epsilon). The first Liapunov constant is

$$\begin{aligned} v_3 = {} & \frac{\alpha}{3\,(r+w^2)\delta^3} - \frac{r+\alpha+w^2}{(r+w^2)\delta^3} - \frac{2\,\alpha w\gamma}{3\,(r+w^2)^2\delta^5} + \frac{2\,w\gamma}{3\,\alpha(r+w^2)^2\delta^3} \\ & + \frac{2\,\gamma^2}{3\,\alpha(r+w^2)^2\delta^5}, \end{aligned}$$

whose numerator $\bar{v}_3$, when expanded, has 34 terms. The numerators $\bar{v}_5, \ldots, \bar{v}_{13}$ of the subsequent 5 Liapunov constants $v_5, \ldots, v_{13}$ consist of $384, 1969, 6616, 17504, 39467$ terms and are of total degrees $18, 28, 38, 48, 58$ in the variables $\eta, r, \alpha, w, \delta, \mu$, respectively.

Let us first take $\eta = 1$ as in [2]. We want to determine real values of r, α and μ such that $a + d = 0$, $a^2 + bc < 0$, $v_3 = v_5 = 0$ but $v_7 \neq 0$, so that the steady state $(0, 0)$ of (3.2) is a focus of order 3. For this purpose, we first compute the plex Gröbner basis $\mathcal{G}$ of $\{f_1, f_2, \delta^2 + f, \bar{v}_3, \bar{v}_5\}$ under the variable order $r \prec \alpha \prec w \prec \delta \prec \mu$ using the `Groebner` package in Maple. It is found that the first polynomial in $\mathcal{G}$ may be factorized as

$$\alpha\, r^7 (4\,r+1)^2 (24\,r-1)^2 (16\,r^2 - 24\,r + 1)\, h, \tag{3.4}$$

where

$$\begin{aligned} h = {} & 1474560\,r^7 - 3997696\,r^6 + 4549632\,r^5 - 4503808\,r^4 \\ & - 4966528\,r^3 - 928256\,r^2 - 24396\,r + 2025. \end{aligned}$$

Since all the physical parameters are required to be positive, we can assume that $\alpha\delta \neq 0$ and $r > 0$, so the factors α, r, $4\,r + 1$ in (3.4) need not be considered. Let z be a new indeterminate and fix the order for the variables as $r \prec \alpha \prec w \prec \delta \prec \mu \prec z$. The reduced plex Gröbner bases of both $\mathcal{G} \cup \{24\,r - 1, z\alpha\delta - 1\}$ and $\mathcal{G} \cup \{16\,r^2 - 24\,r + 1, z\alpha\delta - 1\}$ are equal to $\{1\}$, so the factors $24\,r - 1$ and $16\,r^2 - 24\,r + 1$ need not be considered either. The technique used here to exclude

the case $\alpha\delta = 0$ by introducing z and adding the equation $z\alpha\delta - 1$ is standard and known as *Rabinowich's trick*.

Now compute the plex Gröbner basis $\mathcal{G}^*$ of $\mathcal{G} \cup \{h, z\alpha\delta - 1\}$ with respect to the above variable order: $\mathcal{G}^*$ consists of 6 polynomials, of which the first is h. The set of zeros of $\mathcal{G}^*$ covers all the zeros $(\bar{r}, \bar{\alpha}, \bar{\mu}, \bar{w}, \bar{\delta})$ of $\mathcal{G}$ with $\bar{r} > 0$ and $\bar{\alpha}\bar{\delta} \neq 0$. Isolating the real zeros of $\mathcal{G}^*$ using the Maple package RS (developed by F. Rouillier, see http://fgbrs.lip6.fr/~roullier/Software/RS/), we find that $\mathcal{G}^*$ has 8 real zeros. Among these zeros there is one and only one $(\bar{r}, \bar{\alpha}, \bar{\mu}, \bar{w}, \bar{\delta})$ for which $\bar{r}, \bar{\alpha}, \bar{\mu}, \bar{w}, \bar{\delta}$ are all positive. For this positive zero, we have

$$\begin{aligned} &\bar{r} \approx 0.033247029312587, \quad \bar{\alpha} \approx 0.347417369934422, \\ &\bar{\mu} \approx 1.165669793409291, \quad \bar{w} \approx 0.702121202169318. \end{aligned} \tag{3.5}$$

It may be easily verified by using RS that the real zero $(\bar{r}, \bar{\alpha}, \bar{\mu}, \bar{w})$ satisfies $f < 0$, and that for $\eta = 1$ and $(r, \alpha, \mu) = (\bar{r}, \bar{\alpha}, \bar{\mu})$, $v_3 = 0$, $v_5 = 0$ and $v_7 < 0$. Therefore, the steady state

$$(\bar{x}, \bar{y}) = \left(\frac{\bar{\mu} - \bar{w}}{\bar{\alpha}}, \bar{w}\right) \approx (1.334270049098212, 0.702121202169318) \tag{3.6}$$

is an asymptotically stable focus of order 3 and thus three limit cycles may bifurcate from $(\bar{x}, \bar{y})$ for system (3.1) with small perturbation.

Theorem 2. *For $\eta = 1$, there is one and only one set of positive values $\bar{r}, \bar{\alpha}, \bar{\mu}$ as shown in* (3.5) *for the parameters r, α, μ such that system* (3.1) *has a focus of order* 3. *This focus is located at $(\bar{x}, \bar{y}) = ((\bar{\mu} - \bar{w})/\bar{\alpha}, \bar{w})$ and is asymptotically stable, where $\bar{w}$ is as in* (3.5).

In the next subsection, we will show how to construct a perturbed *polynomial* differential system of (3.2) that has three small-amplitude limit cycles near $(0, 0)$.

3.2 Construction of Limit Cycles

To construct small-amplitude limit cycles, let us consider the following perturbed system of (3.2):

$$\frac{\mathrm{d}X}{\mathrm{d}\tau} = \lambda X + Y + \frac{\delta}{\alpha} Q_1, \quad \frac{\mathrm{d}Y}{\mathrm{d}\tau} = -X + \lambda Y + Q_2, \tag{3.7}$$

where

$$\begin{aligned} Q_1 = {} & \frac{\gamma}{(r + w^2)\delta^3} X^2 - \frac{2\,\alpha w}{(r + w^2)\delta^2} XY - \frac{\alpha(r + \alpha + w^2)}{(r + w^2)\delta^4} X^3 \\ & + \left[\frac{\alpha}{(r + w^2)\delta^3} + \omega\right] X^2 Y, \\ Q_2 = {} & \frac{\gamma}{(r + w^2)\delta^3} X^2 - \frac{2\,\alpha w}{(r + w^2)\delta^2} XY - \left[\frac{\alpha(r + \alpha + w^2)}{(r + w^2)\delta^4} + \xi\right] X^3 \\ & + \left[\frac{\alpha}{(r + w^2)\delta^3} + \theta\right] X^2 Y, \end{aligned}$$

and γ is as in (3.3). The first three Liapunov constants of (3.7) computed by the Epsilon function `miscel[licon]` are as follows:

$$v_3|_{\lambda=0} = \frac{\theta}{3} + \frac{\Gamma_1}{3\,\alpha\delta^5(r+w^2)^2},$$

where Γ_1 consists of 34 terms and is of total degree 8 in the variables $r, \alpha, w, \delta, \mu$;

$$v_5|_{\lambda=0,\,\theta=0} = \frac{\Gamma_2\omega + \Gamma_3\xi + \Gamma_4}{45\,\alpha^3\delta^{11}(r+w^2)^4},$$

where Γ_2, Γ_3, Γ_4 consist of $62, 62, 384$ terms and are of total degrees $15, 15, 12$ in $r, \alpha, w, \delta, \mu$, respectively;

$$v_7|_{\lambda=0,\,\theta=0,\,\omega=0} = \frac{\Gamma_5\xi^2 + \Gamma_6\xi + \Gamma_7}{1890\,\alpha^5\delta^{17}(r+w^2)^6},$$

where Γ_5, Γ_6, Γ_7 consist of $90, 581, 1969$ terms and are of total degrees $25, 22, 19$ in $r, \alpha, w, \delta, \mu$, respectively.

Obviously, if $\lambda = \theta = 0$, then $v_3 = 0$, so $\Gamma_1 = 0$. Thus we may use Γ_1 to reduce the numerator $\bar{v}_5$ of $v_5|_{\lambda=0,\,\theta=0}$ to obtain a remainder R_5 (which consists of 80 terms and is of total degree 20 in $r, \alpha, w, \delta, \mu, \omega, \xi$) such that $\bar{v}_5 = U_5\Gamma_1 + R_5$ for some U_5. Similarly, we can use Γ_1 to reduce the numerator of $v_7|_{\lambda=0,\,\theta=0,\,\omega=0}$ to get a remainder R_7, which consists of 348 terms and is of total degree 20 in $r, \alpha, w, \delta, \mu, \xi$.

Note that $r > 0$, $\alpha > 0$, $\delta > 0$, $w > 0$, $\mu > 0$ and Γ_1 vanishes at any zero of $\mathcal{G}^*$. So one can choose $-1 \ll \theta < 0$, such that $v_3 < 0$. Next, we determine the condition on ω and ξ such that $\mathcal{G}^* = 0$ and $R_5 > 0$. Using the DV package, we compute a minimal discriminant variety $\Delta_1(\omega, \xi)$ of $\mathcal{G}^* = 0, R_5 \neq 0$. By means of partial CAD, we can obtain 18 cells of the two-dimensional real space $\mathbb{R}^2$ of ω, ξ decomposed by $\Delta_1 = 0$. Choosing one sample point in each cell and computing the sign of R_5 at each sample point, we find that under the assumption $r > 0$, $\alpha > 0$, $\delta > 0$, $w > 0$, $\mu > 0$, $|\xi| < 1$, if $\omega < 3/32$, then $R_5 > 0$. Finally, we determine the condition on ξ such that $\mathcal{G}^* = 0$ and $R_7 < 0$. Using the DV package, we compute a minimal discriminant variety $\Delta_2(\xi)$ of $\mathcal{G}^* = 0, R_7 \neq 0$: Δ_2 has 10 real zeros $\xi_1 < \cdots < \xi_{10}$, which divide the real space $\mathbb{R}$ of ξ into 11 intervals. We choose one sample point in each interval, compute the sign of R_7 at each sample point, and thus find that, under the assumption $r > 0$, $\alpha > 0$, $\delta > 0$, $w > 0$, $\mu > 0$, if $\xi_3 < \xi < \xi_{10}$, then $R_7 < 0$, where

$$\xi_3 \in \left[-\frac{2845043675}{4294967296}, -\frac{22760349399}{34359738368}\right], \quad \xi_{10} \in \left[\frac{2379513075}{67108864}, \frac{19036104601}{536870912}\right].$$

Therefore, if we choose sufficiently small values for the perturbation variables successively such that

$$0 < \lambda \ll -\theta \ll |\omega| \ll |\xi| \ll 3/32, \tag{3.8}$$

then the Liapunov constants of (3.7) have alternate signs, i.e.,

$$v_1 = \lambda > 0, \quad v_3 < 0, \quad v_5 > 0, \quad v_7 < 0.$$

In this case, the stability of the focus $(\bar{x}, \bar{y})$ of (3.7) turns over three times and thus three limit cycles in a small neighborhood of $(\bar{x}, \bar{y})$ can bifurcate. The creation of small-amplitude limit cycles by choosing sufficiently small values for the perturbation variables to change the stability of a focus is a typical technique in bifurcation theory (see [18], [28, pp. 214–215], and [29, pp. 272–273]).

Theorem 3. *Three limit cycles may bifurcate in the neighborhood of* $(\bar{x}, \bar{y})$ *for system* (3.1) *with* $(\eta, r, \alpha, \mu) = (1, \bar{r}, \bar{\alpha}, \bar{\mu})$ *and small perturbation, where* $\bar{x}, \bar{y}, \bar{r}, \bar{\alpha}, \bar{\mu}$ *are as in* (3.6) *and* (3.5)*. The perturbed differential system* (3.7) *of* (3.2) *remains polynomial and the perturbation variables* $\lambda, \theta, \omega, \xi$ *may take sufficiently small values satisfying* (3.8)*.*

3.3 Conditions for the Existence of Foci

Condition for (3.1) to have a focus of order 3. It has been shown in Section 3.1 that if $\eta = 1$, then system (3.1) may have a focus of order 3. Now we want to derive the condition on η for system (3.1) to be of center-focus type and have a (positive) focus of order 3. This can be done by deriving the condition on η such that $a + d = 0$, $a^2 + bc < 0$, $v_3 = v_5 = 0$, $v_7 \neq 0$, $r > 0$, $\alpha > 0$, $\mu > 0$, $w > 0$. To do so, we compute a minimal discriminant variety V of $f_1 = 0$, $f_2 = 0$, $f + \delta^2 = 0$, $\bar{v}_3 = 0$, $\bar{v}_5 = 0$, $\bar{v}_7 \neq 0$, $r \neq 0$, $\alpha \neq 0$, $\mu \neq 0$, $w \neq 0$, $\delta \neq 0$ (where $\bar{v}_i$ is the numerator of v_i for $i = 3, 5, 7$) using DV and check the signs of the polynomials at the sample points of the cells of $\mathbb{R}$ decomposed by V. It is then found that for any positive value of η, the semi-algebraic system $a + d = 0$, $a^2 + bc < 0$, $v_3 = 0$, $v_5 = 0$, $v_7 \neq 0$, $r > 0$, $\alpha > 0$, $\mu > 0$, $w > 0$ has at least one real solution for (r, α, μ, w).

Theorem 4. *For any positive value of* η*, there exists at least one set of positive values for the parameters* r, α, μ *such that system* (3.1) *is of center-focus type and has a positive focus of order* 3*.*

Absence of focus of order 4 for (3.1). Finally, we show that system (3.1) does not have any focus of order greater than 3 and thus one cannot construct 4 small-amplitude limit cycles from a focus by perturbation. For this end, we do not take any values for the parameters in (3.1). We want to decide whether there exist real values of r, α, μ, η such that $a + d = 0$, $a^2 + bc < 0$, and $v_3 = v_5 = v_7 = 0$, so that the steady state $(0, 0)$ of (3.2) is a center or a focus of order greater than 3.

Assume that $\alpha\delta\eta r \neq 0$. We compute the plex Gröbner basis $\mathcal{G}$ of $\{f_1, f_2, \delta^2 + f, \bar{v}_3, \bar{v}_5, \bar{v}_7, z\alpha\delta\eta r - 1\}$ under the variable order $r \prec \alpha \prec w \prec \delta \prec \mu \prec \eta \prec z$ using FGb, an efficient Gröbner basis package developed by Faugère (see http://fgbrs.lip6.fr/jcf/Software/FGb): the basis is $\{1\}$. This implies that the system $a + d = 0, a^2 + bc < 0, v_3 = 0, v_5 = 0, v_7 = 0, r \neq 0, \alpha \neq 0, \eta \neq 0$ has no solution for r, α, μ, η, so the following result is proved.

Theorem 5. *There are no nonzero values for the parameters* r, α, μ, η *such that system* (3.1) *is of center-focus type and has a center or a focus of order greater than* 3*.*

It follows from this theorem that for system (3.1) one cannot construct more than 3 small-amplitude limit cycles near a focus by perturbation.

4 Bifurcation and Limit Cycles for Kolmogorov Prey-Predator and Lotka–Volterra Systems

4.1 Cubic Kolmogorov Prey-Predator System

To illustrate the applicability of our approach, in this subsection we discuss briefly the analysis of another two-dimensional biological system, the cubic Kolmogorov prey-predator system constructed and analyzed by Lu and He in [12] using similar algebraic methods. The system is of the form

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= x\,(-2 - a_0 + a_1 + a_0 x - 2\,a_1 x + y + a_1 x^2 + xy), \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= y\,(2 + a_2 - x - y - 2\,a_2 y + a_2 y^2). \end{aligned} \tag{4.1}$$

This system has a positive steady state $(\bar{x}, \bar{y}) = (1, 1)$ and it becomes of center-focus type when $a_0 = 0$. Under the condition $a_0 = 0$, system (4.1) may be transformed by a simple linear transformation $x = Y + X + 1$, $y = -X + 1$ into the following canonical form

$$\begin{aligned} \frac{\mathrm{d}X}{\mathrm{d}t} &= Y - a_2 X^2 - XY + a_2 X^3, \\ \frac{\mathrm{d}Y}{\mathrm{d}t} &= -X - (2 - a_1 - a_2)X^2 + 2\,a_1 XY + (1 + a_1)Y^2 - (1 - a_1 + a_2)X^3 \\ &\quad - (2 - 3\,a_1)X^2 Y - (1 - 3\,a_1)XY^2 + a_1 Y^3. \end{aligned} \tag{4.2}$$

By computing the Liapunov constants v_3, v_5, v_7 of (4.2) and isolating the real zeros of $\{v_3, v_5\}$, we can determine $(a_1, a_2) \approx (0.08020305719, 0.2955574645)$ such that $v_3 = v_5 = 0$ and $v_7 < 0$; in this case the steady state $(1, 1)$ is an asymptotically stable focus of order 3.

We can perturb (4.2) by adding λX to the first equation and λY to the second equation, and substituting a_1 with $a_1 + \omega$ and a_2 with $a_2 + \xi$. Then, compute the Lipunov constants v_3, v_5, v_7 of the perturbed system and use DV to derive conditions on ξ and ω such that $v_3 < 0$, $v_5 > 0$, $v_7 < 0$ as we have done in Section 3.2. In this way, we find that, if the perturbation variables take sufficiently small values such that

$$-0.001 < \xi \ll \omega \ll -\lambda < 0,$$

then the Liapunov constants of the perturbed system have alternate signs, i.e.,

$$v_1 = \lambda > 0, \quad v_3 < 0, \quad v_5 > 0, \quad v_7 < 0.$$

In this case, the stability of the focus $(1, 1)$ turns over three times and thus three limit cycles may bifurcate in a small neighborhood of $(1, 1)$. This conclusion confirms the result given in [12].

4.2 Bifurcation and Limit Cycles for High-Dimensional Systems

Many biological systems are high-dimensional (e.g., of the form (1.1) with $n > 2$). The analysis of bifurcation and limit cycles for such systems is much more difficult. Some theorems and methods such as the generalized Poincaré–Bendixon theorem and the describing function method may be applied for the analysis (see, e.g., [17]), but they are applicable only to certain high-dimensional systems. The methods of center manifold [4,10] and Liapunov–Schmidt reduction [5] allow one to reduce any system of dimension $n > 2$ to a planar system without losing any significant aspect of the dynamic characters. These methods in combination with the method for planar systems presented in Section 2 can be used for the algebraic analysis of bifurcation and limit cycles for high-dimensional systems. Here we use the method of center manifold, which is convenient for our treatment of planar systems using Liapunov constants.

To perform reduction, we need the condition under which the Jacobian matrix $\boldsymbol{J}$ of system (1.1) has a pair of purely imaginary eigenvalues and all the other eigenvalues with negative real parts. Such a condition may be given in terms of the Hurwitz determinants and the constant term of the characteristic polynomial of $\boldsymbol{J}$ according to a simple criterion established by El Kahoui and Weber [7]. The criterion was derived for an arbitrary univariate polynomial A to have one pair of purely imaginary roots and all the other roots with negative real parts by linking the Hurwitz determinants Δ_i of A to the principal subresultant coefficients of A_2 and A_1, where $A_1(\lambda^2) + \lambda A_2(\lambda^2) = A(\lambda)$, and by investigating the behavior of Δ_i in the case where A has symmetric roots. Under the condition determined by using El Kahoui and Weber's criterion [7, Theorem 3.6], we can transform system (1.1) to a system of special form and then reduce the transformed system to a two-dimensional system of center-focus type by using the center manifold theorem. We will see how the reduction proceeds from the example in the following subsection. Bifurcation and limit cycles for the obtained two-dimensional system may be analyzed by using the method explained in Section 2.

4.3 Competitive Three-Dimensional Lotka–Volterra System

In this subsection, we use a competitive three-dimensional Lotka–Volterra system (which models three mutually competing species) to illustrate the analysis of bifurcation and limit cycles by using the center manifold theorem and algebraic methods. The system has the form

$$\begin{aligned}
\frac{\mathrm{d}x_1}{\mathrm{d}t} &= x_1\left[(1-x_1)+(1-x_2)+(1-x_3)\right],\\
\frac{\mathrm{d}x_2}{\mathrm{d}t} &= x_2\left[(1-x_1)+(1-x_2)+2\,(1-x_3)\right],\\
\frac{\mathrm{d}x_3}{\mathrm{d}t} &= x_3\left[\mu_1\,(1-x_1)+\mu_2\,(1-x_2)+3\,(1-x_3)\right],
\end{aligned} \tag{4.3}$$

where μ_1 and μ_2 are two real parameters. This system has been studied in [26,13] (see also [17]) and it has a unique positive steady state $(1, 1, 1)$.

By using the transformation $x = x_1 - 1$, $y = x_2 - 1$, $z = x_3 - 1$ and rewriting the transformed system of (4.3) in the vector form, we obtain

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{D}\boldsymbol{A}\,\boldsymbol{x}, \tag{4.4}$$

where

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 1+x & & \\ & 1+y & \\ & & 1+z \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

The Jacobian matrix of (4.3) at $(1,1,1)$ is the same as $\boldsymbol{A}$. It is not difficult to prove that the characteristic polynomial of $\boldsymbol{A}$ has a negative real root and a pair of purely imaginary roots if and only if

$$\det(\boldsymbol{A}) = (\boldsymbol{A}_{11} + \boldsymbol{A}_{22} + \boldsymbol{A}_{33})\,\mathrm{tr}(\boldsymbol{A}) < 0, \tag{4.5}$$

where

$$\begin{aligned} &\mathrm{tr}(\boldsymbol{A}) = a_{11} + a_{22} + a_{33}, \quad && \boldsymbol{A}_{11} = a_{22}a_{33} - a_{23}a_{32}, \\ &\boldsymbol{A}_{22} = a_{11}a_{33} - a_{13}a_{31}, && \boldsymbol{A}_{33} = a_{22}a_{11} - a_{12}a_{21}. \end{aligned}$$

Condition (4.5) is equivalant to

$$10 - 3\,\mu_2 - 2\,\mu_1 = 0, \quad \mu_2 - \mu_1 < 0.$$

Under this condition and substituting $5 - \frac{3}{2}\mu_2$ for μ_1, we may construct a transformation matrix $\boldsymbol{T}$ according to the method described in [13], and system (4.3) may be transformed by the transformation $\boldsymbol{T}\boldsymbol{x} \to \boldsymbol{x}$ into the following form

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= -\frac{3\,(\mu_2 - 2)}{2\,(\mu_2 - 4)}x - \frac{2\,\mu_2 - 5}{\mu_2 - 4}y + l_1(x, y, z, \mu_2), \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= -\frac{(\mu_2 - 10)(\mu_2 - 2)}{4\,(\mu_2 - 4)}x + \frac{3\,(\mu_2 - 2)}{2\,(\mu_2 - 4)}y + l_2(x, y, z, \mu_2), \\ \frac{\mathrm{d}z}{\mathrm{d}t} &= -5\,z + l_3(x, y, z, \mu_2), \end{aligned} \tag{4.6}$$

where $l_i(x, y, z, \mu_2)$ are the terms nonlinear in x, y, z. Then by the center manifold theorem [10, p. 152], we may analyze the bifurcation and limit cycles for system (4.6) on a two-dimensional center manifold. The center manifold up to approximation of order k can be described as

$$z = \Psi(x, y) = \sum_{i=2}^{k} \sum_{j=0}^{i} c_{ij} x^{i-j} y^j + \text{h.o.t.}, \tag{4.7}$$

where h.o.t. denotes the terms of higher order ($> k$). It follows that

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \frac{\partial \Psi}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial \Psi}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t}. \tag{4.8}$$

To compute for instance the first two Liapunov constants of the two-dimensional system (to be determined), we need to consider the center manifold up to quartic approximation. Substituting

$$\frac{\mathrm{d}z}{\mathrm{d}t} = -5\,z + l_3(x, y, z, \mu_2) \quad \text{with} \quad z = \sum_{i=2}^{4} \sum_{j=0}^{i} c_{ij} x^{i-j} y^j$$

into (4.8) and comparing the coefficients of the two sides of the two equations in $x, y, \mathrm{d}x/\mathrm{d}t, \mathrm{d}y/\mathrm{d}t$, we obtain a system of linear equations in c_{ij} ($i = 2, \ldots, 4$, $j = 0, \ldots, 2$). Then c_{ij} may be found as rational functions $\bar{c}_{ij}$ of μ_2, which are quite complicated (for example, the numerator and the denominator of $\bar{c}_{20}$ both consist of 29 terms and are of degree 28 in μ_2). Substituting $z = \Psi(x, y)|_{c_{ij}=\bar{c}_{ij}}$ into the first two equations in (4.6) results in a two-dimensional system of center-focus type. This system may be further transformed to the canonical form by the linear transformation

$$x = \frac{4\,(\mu_2 - 4)}{(\mu_2 - 10)(\mu_2 - 2)}\,Y - \frac{6}{(\mu_2 - 10)\,\delta}\,X, \quad y = -\frac{1}{\delta}\,X, \quad t = \frac{\tau}{\delta},$$

where $\delta = \sqrt{-\mu_2/2 + 1}$. Then by using the Epsilon function `miscel[licon]`, we can compute the first two Liapunov constants v_3 and v_5 of the two-dimensional system in the canonical form.[1] The numerators $\bar{v}_3, \bar{v}_5$ of v_3, v_5 and the denominator of v_5 consist of 44, 197 and 35 terms and are of total degrees 16, 39 and 45 in δ, μ_2, respectively. Computation using RS shows that $\{\bar{v}_3, \delta^2 + \mu_2/2 - 1\}$ has 7 isolated real zeros for (μ_2, δ). Let the physical parameters μ_2 and $\mu_1 = 5 - \frac{3}{2}\mu_2$ be assumed positive. Then there are two real values $\bar{\mu}_2'$ and $\bar{\mu}_2''$ for μ_2 such that $\det(\boldsymbol{A}) < 0$, where

$$\bar{\mu}_2' \in \left[\frac{12946800109}{8589934592}, \frac{25893600219}{17179869184}\right] = I, \quad \bar{\mu}_2'' = \frac{8}{5}.$$

It is easy to verify by using RS that $\bar{v}_3 = 0$, $v_5 > 0$ at $\bar{\mu}_2'$, $v_3 < 0$ at the right end of I, and the denominator of v_3 has no real root in I. Therefore, the steady state $(1, 1, 1)$ is unstable and one can perturb $\bar{\mu}_2'$ slightly to $\bar{\mu}_2^*$ for μ_2 such that $v_3 < 0$, while $v_5 > 0$ remains true, so that one limit cycle bifurcates. Next, perturb $5 - \frac{3}{2}\bar{\mu}_2^*$ slightly to $\bar{\mu}_1^*$ for μ_1 such that $v_3 < 0$, $v_5 > 0$ remain true, while the real part of the two conjugate complex eigenvalues of the Jacobian matrix of the two-dimensional system becomes positive. Then the second limit cycle bifurcates. Our construction of two limit cycles here is similar to the constructions in [13].

For the other value $\bar{\mu}_2'' = 8/5$ of μ_2, $\det(\boldsymbol{A}) < 0$ and $v_3 = v_5 = 0$, so the steady state is likely to be a center (see [26]).

[1] Most of the computations discussed in this paper were performed step by step on a laptop T2400 with 2 CPUs, 1.83 GHz and 987 MHz, and 2 GB RAM. Some of them were verified by using different software tools. Each successful step of computation usually takes no more than a few minutes. However, the computation of v_5 here requires nearly one hour.

Analysis of bifurcation and limit cycles for high-dimensional systems by using the method of center manifold involves complicated calculations and the produced planar systems are usually large, so the Liapunov constants become very large and unmanageable. For example, to construct three small-amplitude limit cycles for a three-dimensional Lotka–Volterra system one needs to compute the first three Liapunov constants and thus the center manifold needs to be approximated up to order 6. In this case, the computation of the Liapunov constants of the produced planar system even becomes rather difficult.

5 Concluding Remarks

In this paper, we have shown how to analyze bifurcation and limit cycles for two biological systems of dimension 2 and another of dimension 3 using an algebraic approach based on the methods of triangular decomposition, Gröbner bases, discriminant varieties, real solution classification, and quantifier elimination by partial CAD. Several efficient software packages including Epsilon, FGb, RS, DV, and DISCOVERER in which some of these algebraic methods are implemented have been used for the involved symbolic computations with semi-algebraic systems. The analysis of bifurcation and limit cycles is presented in detail for the cubic self-assembling micelle system with chemical sinks and briefly for the cubic Kolmogorov prey-predator system, both two-dimensional. It is proved that the self-assembling micelle system may have a focus of order 3, from which three limit cycles can be constructed by small perturbation, but this system cannot have a focus of order greater than 3. Bifurcation analysis for the competitive Lotka–Volterra system is carried out by using the method of center manifold that reduces the three-dimensional system to a two-dimensional one. Two limit cycles are then constructed from the determined focus of order 2 by small perturbation.

The investigations described in this paper illustrate the applicability and limitation of our algebraic approach for the analysis of bifurcation and limit cycles for biological systems. This approach uses exact symbolic computation and ensures that all the obtained results are mathematically rigorous. It is a useful alternative to the experimental approach based on numerical simulation and visualization.

Although the algebraic methods underlying our approach have been well developed and are powerful, it is well known that symbolic computations involved in these methods are very heavy in general. How to improve the methods by introducing new and specialized algebraic techniques, how to use them to analyze bifurcation and limit cycles for high-dimensional biological systems more effectively, and how to deal with biological systems of the form (1.1) with p_i being rational or other functions are some of the questions that remain for future research.

References

1. Andronov, A.A., Leontovich, E.A., Gordon, I.I., Maier, A.G.: Theory of Bifurcations of Dynamic Systems on a Plane. Israel Program of Scientific Translations, Jerusalem (1971)

2. Ball, R., Haymet, A.D.J.: Bistability and hysteresis in self-assembling micelle systems: Phenomenology and deterministic dynamics. Phys. Chem. Chem. Phys. 3, 4753–4761 (2001)
3. Buchberger, B.: Gröbner bases: An algorithmic method in polynomial ideal theory. In: Bose, N.K. (ed.) Multidimensional Systems Theory, pp. 184–232. Reidel, Dordrecht (1985)
4. Carr, J.: Applications of Center Manifold Theory. Springer, New York (1981)
5. Chow, S.-N., Hale, J.K.: Methods of Bifurcation Theory. Springer, New York (1982)
6. Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. J. Symb. Comput. 12, 299–328 (1991)
7. El Kahoui, M., Weber, A.: Deciding Hopf bifurcations by quantifier elimination in a software-component architecture. J. Symb. Comput. 30, 161–179 (2000)
8. Farr, W.W., Li, C., Labouriau, I.S., Langford, W.F.: Degenerate Hopf bifurcation formulas and Hilbert's 16th problem. J. Math. Anal. 20(1), 13–30 (1989)
9. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F_4). J. Pure Appl. Algebra 139, 61–88 (1999)
10. Kuznetsov, Y.A.: Elements of Applied Bifurcation Theory, 2nd edn. Springer, New York (1998)
11. Lazard, D., Rouillier, F.: Solving parametric polynomial systems. J. Symb. Comput. 42, 636–667 (2007)
12. Lu, Z., He, B.: Multiple stable limit cycles for a cubic Kolmogorov prey-predator system. J. Eng. Math. 18(4), 115–117 (2001)
13. Lu, Z., Luo, Y.: Two limit cycles in three-dimensional Lotka–Volterra systems. Comput. Math. Appl. 44(1/2), 51–66 (2002)
14. Miller, R.K., Michel, A.N.: Ordinary Differential Equations. Academic Press, New York, London (1982)
15. Niu, W.: Application of quantifier elimination and discriminant varieties to stability analysis of biological systems. In: Wang, D., Zheng, Z. (eds.) Proceedings MACIS 2006, pp. 243–253. Beihang University, China (2006)
16. Niu, W., Wang, D.: Algebraic approaches to stability analysis of biological systems. Math. Comput. Sci. 1(3), 507–539 (2008)
17. Shahruz, S.M., Kalkin, D.A.: Limit cycle behavior in three- or higher-dimensional non-linear systems: The Lotka–Volterra example. J. Sound Vibration 246(2), 379–386 (2001)
18. Shi, S.: A concrete example of the existence of four limit cycles for plane quadratic systems. Sci. Sinica (A) 23, 153–158 (1980)
19. Wang, D.: A class of cubic differential systems with 6-tuple focus. J. Diff. Eqns. 87, 305–315 (1990)
20. Wang, D.: Mechanical manipulation for a class of differential systems. J. Symb. Comput. 12, 233–254 (1991)
21. Wang, D.: Elimination Methods. Springer, Wien, New York (2001)
22. Wang, D.: Elimination Practice: Software Tools and Applications. Imperial College Press, London (2004)
23. Wang, D., Xia, B.: Stability analysis of biological systems with real solution classification. In: Kauers, M. (ed.) Proceedings ISSAC 2005, pp. 354–361. ACM Press, New York (2005)
24. Wang, D., Xia, B.: Algebraic analysis of stability for some biological systems. In: Anai, H., Horimoto, K. (eds.) Proceedings AB 2005, pp. 75–83. Universal Academy Press, Tokyo (2005)
25. Wu, W.-t.: Mathematics Mechanization. Science Press/Kluwer, Beijing (2000)

26. Xiao, D., Li, W.: Limit cycles for the competitive three dimensional Lotka–Volterra system. J. Diff. Eqns. 164, 1–15 (2000)
27. Yang, L., Xia, B.: Real solution classifications of parametric semi-algebraic systems. In: Dolzmann, A., Seidl, A., Sturm, T. (eds.) Algorithmic Algebra and Logic—Proceedings A3L 2005, pp. 281–289. Herstellung and Verlag, Norderstedt (2005)
28. Ye, Y., Cai, S., Chen, L., Huang, K., Luo, D., Ma, Z., Wang, E., Wang, M., Yang, X.: Theory of Limit Cycles. Amer. Math. Soc., Providence (1986)
29. Zhang, Z., Ding, T., Huang, W., Dong, Z.: Qualitative Theory of Differential Equations. Amer. Math. Soc., Providence (1992)

The Smallest Multistationary Mass-Preserving Chemical Reaction Network

Anne Shiu

Dept. of Mathematics, University of California,
Berkeley, CA 94720-3840, USA

Abstract. Biochemical models that exhibit bistability are of interest to biologists and mathematicians alike. Chemical reaction network theory can provide sufficient conditions for the existence of bistability, and on the other hand can rule out the possibility of multiple steady states. Understanding small networks is important because the existence of multiple steady states in a subnetwork of a biochemical model can sometimes be lifted to establish multistationarity in the larger network. This paper establishes the smallest reversible, mass-preserving network that admits bistability and determines the semi-algebraic set of parameters for which more than one steady state exists.

Keywords: Chemical reaction network, bistability.

1 Introduction

Bistable biochemical models are often presented as the possible underpinnings of chemical switches [2,17]. Systematic study of mass-action kinetics models–which *a priori* may or may not admit multiple steady states–constitutes chemical reaction network theory (CRNT), a subject pioneered by Horn, Jackson, and Feinberg [13,16]. Certain classes of networks, such as those of deficiency zero, do not exhibit multistationarity or other strange behaviors. A generalization of deficiency-zero systems is the class of *toric dynamical systems* which have a unique steady state [5]. See also the recent work of Craciun and Feinberg for additional conditions that rule out multistationarity [6,7].

On the other hand, there are conditions that are sufficient for establishing whether a network supports multiple steady states. The CRNT Toolbox developed by Feinberg and improved by Ellison implements the Deficiency One and Advanced Deficiency Algorithms [9,12]; this software is available online [10]. For a large class of systems, the CRNT Toolbox either provides a witness for multistationarity or concludes that it is impossible. For systems for which the CRNT Toolbox is inconclusive, see the approach of Conradi *et al.* [4]. Related work includes an algebraic approach that determines the full set of parameters for which a system is multistationary; a necessary and sufficient condition for multistationarity is the existence of a non-trivial sign vector in the intersection of two subsets of Euclidean space [3].

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 172–184, 2008.

To model biological processes, one typically reverse-engineers a system of nonlinear differential equations that exhibit specific dynamical behavior, such as bistability or oscillations, observed in experiments. For example, Segel proposes a small immune network consisting only two cell types, which has three stable steady states, corresponding to "normal," "vaccinated," and "diseased" states [19]. Similarly, the Brusselator is a mass-action kinetics network with a stable limit cycle [11].

This paper focuses on the smallest mass-action kinetics networks that admit multiple steady states. Section 2 provides an introduction to chemical reaction network theory. A special network called the *Square* is shown in Section 3 to be a smallest reversible multistationary chemical reaction network. Sections 4 and 5 determine precisely which parameters of the Square give rise to multiple steady states.

2 Chemical Reaction Network Theory

We now give an introduction to chemical reaction network theory. Before giving precise definitions, we present an intuitive example that illustrates how a chemical reaction network gives rise to a dynamical system. An example of a *chemical reaction*, as it usually appears in the literature, is the following:

$$A + B \xrightarrow{\kappa} 3A + C$$

In this reaction, one unit of chemical *species* A and one of B react (at reaction rate κ) to form three units of A and one of C. The concentrations c_A, c_B, and c_C will change in time as the reaction occurs. Under the assumption of *mass-action kinetics*, species A and B react at a rate proportional to the product of their concentrations, where the proportionality constant is the rate constant κ. Noting that the reaction yields a net change of two in the amount of A, we obtain the first differential equation in the following system:

$$\begin{aligned} \frac{d}{dt}c_A &= 2\kappa c_A c_B, \\ \frac{d}{dt}c_B &= -\kappa c_A c_B, \\ \frac{d}{dt}c_C &= \kappa c_A c_B. \end{aligned}$$

The other two equations arise similarly. Next we include the reverse reaction and switch from additive to multiplicative notation to highlight the monomials that appear in our differential equations; the *chemical reaction networks* in this paper will appear with the following notation:

$$c_A c_B \underset{\kappa'}{\overset{\kappa}{\rightleftarrows}} c_A^3 c_C$$

This network defines differential equations that are each a sum of the monomial contribution from the reactant of each chemical reaction in the network:

$$\begin{aligned}\frac{d}{dt}c_A &= 2\kappa c_A c_B - 2\kappa' c_A^3 c_C,\\ \frac{d}{dt}c_B &= -\kappa c_A c_B + \kappa' c_A^3 c_C,\\ \frac{d}{dt}c_C &= \kappa c_A c_B - \kappa' c_A^3 c_C.\end{aligned}$$

The recipe for obtaining these differential equations from a chemical reaction network easily generalizes from this example. However, in order to display the linearity hidden in these non-linear equations, the equations will appear in a different but equivalent form in (1) below.

We now establish the notation for this paper, following [5]. A *chemical reaction network* is a finite directed graph whose vertices are labeled by monomials and whose edges are labeled by parameters. Specifically, the digraph is denoted $G = (V, E)$, with vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E \subseteq \{(i, j) \in V \times V : i \neq j\}$. The vertex i of G represents the ith chemical complex and is labeled by the monomial

$$c^{y_i} = c_1^{y_{i1}} c_2^{y_{i2}} \cdots c_s^{y_{is}}.$$

This yields $Y = (y_{ij})$, an $n \times s$-matrix of non-negative integers. The unknowns $c_1, c_2, \ldots, c_s$ represent the concentrations of the s species in the network, and we regard them as functions $c_i(t)$ of time t. The monomial labels form the entries in the following row vector:

$$\Psi(c) = \left(c^{y_1},\ c^{y_2},\ \ldots,\ c^{y_n}\right).$$

A network is said to be *mass-preserving* if all monomials c^{y_i} have the same degree. Each directed edge $(i, j) \in E$ is labeled by a positive parameter κ_{ij} which represents the rate constant in the reaction from the i-th chemical complex to the j-th chemical complex. A network is *reversible* if the graph G is undirected, in which case each undirected edge has two labels κ_{ij} and κ_{ji}. Let A_κ denote the negative of the *Laplacian* of the digraph G. In other words A_κ is the $n \times n$-matrix whose off-diagonal entries are the κ_{ij} and whose row sums are zero. Mass-action kinetics specified by the digraph G is the dynamical system defined by

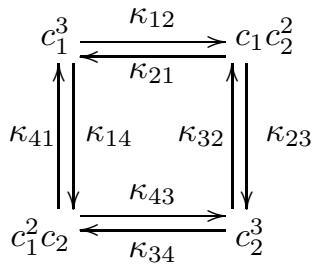
$$\frac{dc}{dt} = \Psi(c) \cdot A_\kappa \cdot Y. \tag{1}$$

By decomposing the mass-action equations in this way, we see that they are linear in the κ_{ij} by way of the matrix A_κ. A *steady state* (or equilibrium) is a positive concentration vector $c \in \mathbb{R}^s_{>0}$ at which the equations (1) vanish. These equations remain in the (stoichiometric) subspace S spanned by the vectors $y_i - y_j$ (where (i, j) is an edge of G). In the earlier example, $y_i - y_j = (-2, -1, 1)$, meaning that whenever a reaction occurs, two units of A and one of B are lost, while one unit of C is formed (or vice-versa). Therefore, a trajectory $c(t)$ beginning at a

positive vector $c(0) = c^0$ remains in the *invariant polyhedron* $P := (c^0 + S) \cap \mathbb{R}^s_{\geq 0}$. *Multistationarity* refers to the existence of more than one steady state in some invariant polyhedron. A chemical reaction network may admit multistationarity for all, some, or no choices of positive parameters κ_{ij}.

Horn initiated the investigation of small chemical reaction networks by enumerating networks comprised of "short complexes," those whose corresponding monomials c^y have degree at most two [14,15]. Networks that consist of at most three short complexes do not permit multiple steady states.

The next section establishes that the following graph, which we call the *Square*, depicts a smallest reversible multistationary chemical reaction network:



In the horizontal reactions, two units of species one are transformed into two of species two (or vice-versa), while a third unit remains unchanged by the reaction. In the vertical reactions, only one is transformed.

The Square appeared in non-reversible form as networks 7-3 in [16] and 4.2 in [11]. The matrices whose product defines the dynamical system (1) follow:

$$\Psi(c) \quad = \quad \left(c_1^3,\ c_1c_2^2,\ c_2^3,\ c_1^2c_2\right),$$

$$A_\kappa \quad = \quad \begin{pmatrix} -\kappa_{12}-\kappa_{14} & \kappa_{12} & 0 & \kappa_{14} \\ \kappa_{21} & -\kappa_{21}-\kappa_{23} & \kappa_{23} & 0 \\ 0 & \kappa_{32} & -\kappa_{32}-\kappa_{34} & \kappa_{34} \\ \kappa_{41} & 0 & \kappa_{43} & -\kappa_{41}-\kappa_{43} \end{pmatrix},$$

$$Y \quad = \quad \begin{pmatrix} 3 & 0 \\ 1 & 2 \\ 0 & 3 \\ 2 & 1 \end{pmatrix}.$$

There may be two or even three steady states in each invariant polyhedron P; Example 1 in the next section provides a choice of positive rate constants κ_{ij} that give rise to three steady states. Sections 4 and 5 determine precisely which parameters give rise to two steady states and which yield three. Moreover, we compute this semi-algebraic parametrization for all networks on the same four vertices as the Square, in other words, networks with complexes c_1^3, $c_1c_2^2$, c_2^3, and $c_1^2c_2$. The parametrization is captured in Table 1 and can be computed "by hand," but larger systems may require techniques of computational real algebraic geometry [1]. For example, our problem of classifying parameters according to number of steady states is labeled as Problem P2 in [21], where it is addressed with computer algebra methods.

3 The Smallest Multistationary Network

Following equation (7) of [5], the Matrix-Tree Theorem defines the following polynomials in the rate constants of the Square:

$$\begin{aligned} K_1 &= \kappa_{23}\kappa_{34}\kappa_{41} + \kappa_{21}\kappa_{34}\kappa_{41} + \kappa_{21}\kappa_{32}\kappa_{41} + \kappa_{21}\kappa_{32}\kappa_{43}, \\ K_2 &= \kappa_{14}\kappa_{32}\kappa_{43} + \kappa_{12}\kappa_{34}\kappa_{41} + \kappa_{12}\kappa_{32}\kappa_{41} + \kappa_{12}\kappa_{32}\kappa_{43}, \\ K_3 &= \kappa_{14}\kappa_{23}\kappa_{43} + \kappa_{14}\kappa_{21}\kappa_{43} + \kappa_{12}\kappa_{23}\kappa_{41} + \kappa_{12}\kappa_{23}\kappa_{43}, \\ K_4 &= \kappa_{14}\kappa_{23}\kappa_{34} + \kappa_{14}\kappa_{21}\kappa_{34} + \kappa_{14}\kappa_{21}\kappa_{32} + \kappa_{12}\kappa_{23}\kappa_{34}. \end{aligned}$$

Theorem 7 of [5] provides an ideal M_G that is toric in these K_i coordinates, and the variety of M_G is the moduli space of toric dynamical systems on the Square. In this case, the ideal M_G is the *twisted cubic curve* in the K_i coordinates, generated by the 2×2-minors of the following matrix:

$$\begin{pmatrix} K_1 & K_2 & K_4 \\ K_4 & K_3 & K_2 \end{pmatrix}. \tag{2}$$

Theorem 7 of [5] says that for a given choice of positive rate constants κ_{ij}, the equations (1) define a toric dynamical system if and only if the minors of the matrix (2) vanish. In general the codimension of M_G is the *deficiency* of a network; see Theorem 9 of [5]. Here the deficiency is two. Recall that a *toric dynamical system* is a dynamical system (1) for which the algebraic equations $\Psi(c) \cdot A_\kappa = 0$ admit a strictly positive solution $c^* \in \mathbb{R}^s_{>0}$; this solution is called a complex balancing steady state [16]. In this case there is a unique steady state in each invariant polyhedron P. Toric dynamical systems exhibit further nice properties; for details, see [5,13,16].

It is no coincidence that the original monomials of the Square, namely c_1^3, $c_1c_2^2$, c_2^3, $c_1^2c_2$, parametrize the twisted cubic curve. In fact, the following general result follows from Theorem 9 in [5].

Proposition 1. *Assume that a chemical reaction network G is strongly connected and all monomials c^{y_i} have the same total degree. Then the toric variety parametrized by $\Psi(c)$ coincides with the variety of M_G.*

For the Square, each one-dimensional invariant polyhedron P is defined by some positive concentration total $T = c_1 + c_2$. The steady states in P correspond precisely to the positive roots of the following cubic polynomial:

$$p_S(x) = (-2\kappa_{12} - \kappa_{14})x^3 + (\kappa_{41} - \kappa_{43})x^2 + (2\kappa_{21} - \kappa_{23})x + (\kappa_{32} + 2\kappa_{34});$$

this polynomial arises by substituting $x := c_1/c_2$ into the equation $dc_1/dt = -dc_2/dt$. From this point of view, we reach some immediate conclusions. First, the *algebraic degree* of this system is three, which bounds the number of steady states. Second, the number of steady states and their stability depend only on the rate parameters κ_{ij}, and not on the invariant polyhedron P or equivalently the choice of total concentration T. Also, by noting that $p_S(x)$ is positive at

$x = 0$ and is negative for large x, we see that the Square admits at least one steady state for any choice of rate constants. Recall that the discriminant of a univariate polynomial f is a polynomial that vanishes precisely when f has a multiple root over the complex numbers [20]. `Maple` computes the discriminant of p_S to be the following polynomial:

$$
\begin{aligned}
&- 108\kappa_{12}^2\kappa_{32}^2 - 432\kappa_{12}^2\kappa_{32}\kappa_{34} - 432\kappa_{12}^2\kappa_{34}^2 - 108\kappa_{12}\kappa_{14}\kappa_{32}^2 \\
&- 432\kappa_{12}\kappa_{14}\kappa_{32}\kappa_{34} - 432\kappa_{12}\kappa_{14}\kappa_{34}^2 + 64\kappa_{12}\kappa_{21}^3 - 96\kappa_{12}\kappa_{21}^2\kappa_{23} + 48\kappa_{12}\kappa_{21}\kappa_{23}^2 \\
&- 72\kappa_{12}\kappa_{21}\kappa_{32}\kappa_{41} + 144\kappa_{12}\kappa_{21}\kappa_{32}\kappa_{43} - 144\kappa_{12}\kappa_{21}\kappa_{34}\kappa_{41} + 288\kappa_{12}\kappa_{21}\kappa_{34}\kappa_{43} \\
&- 8\kappa_{12}\kappa_{23}^3 + 36\kappa_{12}\kappa_{23}\kappa_{32}\kappa_{41} - 72\kappa_{12}\kappa_{23}\kappa_{32}\kappa_{43} + 72\kappa_{12}\kappa_{23}\kappa_{34}\kappa_{41} \\
&- 144\kappa_{12}\kappa_{23}\kappa_{34}\kappa_{43} - 27\kappa_{14}^2\kappa_{32}^2 - 108\kappa_{14}^2\kappa_{32}\kappa_{34} - 108\kappa_{14}^2\kappa_{34}^2 + 32\kappa_{14}\kappa_{21}^3 \\
&- 48\kappa_{14}\kappa_{21}^2\kappa_{23} + 24\kappa_{14}\kappa_{21}\kappa_{23}^2 - 36\kappa_{14}\kappa_{21}\kappa_{32}\kappa_{41} + 72\kappa_{14}\kappa_{21}\kappa_{32}\kappa_{43} \\
&- 72\kappa_{14}\kappa_{21}\kappa_{34}\kappa_{41} + 144\kappa_{14}\kappa_{21}\kappa_{34}\kappa_{43} - 4\kappa_{14}\kappa_{23}^3 + 18\kappa_{14}\kappa_{23}\kappa_{32}\kappa_{41} \\
&- 36\kappa_{14}\kappa_{23}\kappa_{32}\kappa_{43} + 36\kappa_{14}\kappa_{23}\kappa_{34}\kappa_{41} - 72\kappa_{14}\kappa_{23}\kappa_{34}\kappa_{43} + 4\kappa_{21}^2\kappa_{41}^2 \\
&- 16\kappa_{21}^2\kappa_{41}\kappa_{43} + 16\kappa_{21}^2\kappa_{43}^2 - 4\kappa_{21}\kappa_{23}\kappa_{41}^2 + 16\kappa_{21}\kappa_{23}\kappa_{41}\kappa_{43} - 16\kappa_{21}\kappa_{23}\kappa_{43}^2 \\
&+ \kappa_{23}^2\kappa_{41}^2 - 4\kappa_{23}^2\kappa_{41}\kappa_{43} + 4\kappa_{23}^2\kappa_{43}^2 - 4\kappa_{32}\kappa_{41}^3 + 24\kappa_{32}\kappa_{41}^2\kappa_{43} - 48\kappa_{32}\kappa_{41}\kappa_{43}^2 \\
&+ 32\kappa_{32}\kappa_{43}^3 - 8\kappa_{34}\kappa_{41}^3 + 48\kappa_{34}\kappa_{41}^2\kappa_{43} - 96\kappa_{34}\kappa_{41}\kappa_{43}^2 + 64\kappa_{34}\kappa_{43}^3.
\end{aligned}
$$

As p_S is cubic and has at least one positive root, its discriminant is negative if and only if p_S has one real root and one pair of complex conjugate roots; in this case, the Square has precisely one steady state. When the discriminant is non-negative, the system may admit one, two, or three steady states; we analyze this case fully in the next section.

Example 1. Consider the following rate constants for the Square:

$$(\kappa_{12},\ \kappa_{14},\ \kappa_{21},\ \kappa_{23},\ \kappa_{32},\ \kappa_{34},\ \kappa_{41},\ \kappa_{43}) \quad = \quad (1/4,\ 1/2,\ 1,\ 13,\ 1,\ 2,\ 8,\ 1).$$

This yields $p_S(x) = -x^3 + 6x^2 - 11 + 6$, which has three positive roots: $x = 1$, 2, and 3. This is an instance of bistability; it is easy to determine that $x = 1$ and $x = 3$ correspond to *stable* steady states, while the third is unstable. In the next section we determine the conditions for an arbitrary vector of rate constants to admit one, two, or three steady states.

Recalling the definitions given earlier, the Square has the following properties: the number of complexes is $n = 4$, the number of connected components of G is $l = 1$, the number of species is $s = 2$, and the dimension of any invariant polyhedron is $\sigma = 1$. The main result of this section states that this network is minimal with respect to each of these four parameters.

Theorem 1. *The Square is a smallest multistationary, mass-preserving, reversible chemical reaction network with respect to each of the following parameters: the number of complexes, the number of connected components of G, the number of species, and the dimension of an invariant polyhedron.*

Proof. First $l = 1$ and $\sigma = 1$ are clearly minimal. Next any mass-preserving system with $n \leq 2$ or $s = 1$ has no reactions or has deficiency zero. Finally, an $n = 3$ system has deficiency zero or one; in the deficiency one case, the Deficiency One Theorem of Feinberg rules out the possibility of multistationarity [12]. □

Among all mass-preserving multistationary systems that share these four minimal parameters, the Square is distinguished because its monomials are of minimal degree. A connected network of lower degree would consist of at most three of Horn's "short" complexes [14].

We now discuss the possible connection of the Square to biology by comparing it to the following simple network:

$$\begin{aligned} c_x c_y &\leftrightarrows c_y^2 \\ c_x &\leftrightarrows c_y. \end{aligned} \tag{3}$$

Network (3) is a modified version of the following molecular switch mechanism proposed by Lisman [18]:

$$\begin{aligned} c_x c_y &\leftrightarrows c_{xy} \longrightarrow c_y^2 \\ c_y c_p &\leftrightarrows c_{yp} \longrightarrow c_x c_p. \end{aligned}$$

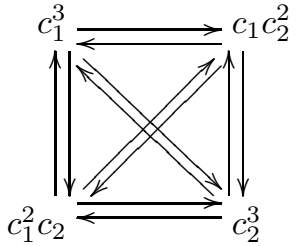
Here x denotes a kinase in an inactive state, y is the active version, and p is a phosphatase. In the first reactions, y catalyzes the phosphorylation of x, turning x into y; the second reactions correspond to dephosphorylation. By skipping the binding steps, making all reactions reversible, and noting that removing p effectively scales the second reaction rate constant, we obtain the network (3). The reactions of (3) are similar to $c_1^2 c_y \leftrightarrows c_2^3$ and $c_1^3 \leftrightarrows c_2^3$ are reactions in the generalization of the Square network examined in the next section; this suggests the possible biological relevance of the reactions of the Square. For example $c_1^2 c_2 \longrightarrow c_2^3$ can be viewed as a reaction in which species two catalyzes the reaction $c_1^2 \longrightarrow c_2^2$. Such a positive feedback loop–in which a high amount of some species y encourages the further production of the same species–occurs in biological settings. For example, the recent work of Dentin *et al.* finds that high glucose levels in diabetic mice promote further glucose production in the liver, which is triggered by the binding of glucose (which we may view as y) to the transcription factor CREB (x) [8].

This paper focuses on the Square and more generally, the networks that share the same complexes as the Square. In the following section, we shall determine which of these are bistable. The one with the fewest edges is the only one with two connected components rather than one, and is featured in the last section.

4 Parametrizing Multistationarity

The aim of this section is similar to that of Conradi *et al.* [3], which determined the full set of parameters that give rise to multistationarity for a biochemical model describing a single layer of a MAPK cascade. However we additionally

determine the precise number of steady states: zero, one, two, or three, and determine their stability. The family of networks we consider are those that have the same four complexes as the Square. In other words, we classify subnetworks of the complete network depicted here:



Each of the twelve rate constants κ_{ij} is permitted to be zero, which defines the parameter space $\mathbb{R}^{12}_{\geq 0}$ of dynamical systems. The main result of this section is summarized in Table 1, which is the semi-algebraic decomposition of the twelve-dimensional parameter space according to the number of steady states of the dynamical system. The conditions listed there make use of certain polynomials in the rate constants, including the signed coefficients of the polynomial p:

$$\begin{aligned} S_0 &= 2\kappa_{12} + 3\kappa_{13} + \kappa_{14}, \\ S_1 &= \kappa_{41} - \kappa_{42} - 2\kappa_{43}, \\ S_2 &= -2\kappa_{21} + \kappa_{23} - \kappa_{24}, \\ S_3 &= 3\kappa_{31} + \kappa_{32} + 2\kappa_{34}, \end{aligned}$$

where p generalizes the polynomial p_S from the Square:

$$p(x) = -S_0x^3 + S_1x^2 - S_2x + S_3.$$

We now derive the entries of Table 1 for those networks without boundary steady states (this includes the case of the Square). These cases are precisely the ones in which $S_0 > 0$ and $S_3 > 0$. Our approach is simply to determine the conditions on the coefficients of p for the polynomial to have one, two, or three positive roots.

In this twelve-parameter case, the discriminant of p is a homogeneous degree-four polynomial with 113 terms. For the same reason as that for the Square, there is one steady state when the discriminant is negative. Now assume that the discriminant is non-negative. Then p has three real roots, counting multiplicity; recall that the positive ones correspond to the steady states of the chemical reaction network. Now the constant term of a monic cubic polynomial is the negative of the product of its roots, so by examining the sign of the constant term of p, we conclude that p has either one positive root and two negative roots, or three positive roots. Continuing the sign analysis with the other coefficients of p, we conclude that there are three positive roots if and only if $S_1 > 0$ and $S_2 > 0$. We proceed by distinguishing between the cases when the discriminant is positive or zero. If the discriminant is positive, then we have derived criteria for having one or three steady states; this is because the roots of p are distinct.

Table 1. Classification of dynamical systems arising from non-trivial (having at least one reaction) networks with complexes c_1^3, $c_1c_2^2$, c_2^3, $c_1^2c_2$. Listed are the number of steady states and the number of steady states that are stable. The discriminant of p is denoted by D. The signed coefficients of p are denoted by S_0, S_1, S_2, and S_3. The triple root condition consists of the equations (4).

Condition	Steady states	Stable states
$D < 0$ and $S_0S_3 = 0$	0	0
$D < 0$ and *else*	1	1
$D > 0$ and $S_0, S_1, S_2, S_3 > 0$	3	2
$D > 0$ and $S_0, S_1, S_2 > 0$ and $S_3 = 0$	2	1
$D > 0$ and $S_1, S_2, S_3 > 0$ and $S_0 = 0$	2	1
$D > 0$ and $S_0 = S_3 = 0$ and $S_1S_2 < 0$	0	0
$D > 0$ and *else*	1	1
$D = 0$ and $S_0, S_1, S_2, S_3 > 0$ and triple root condition	1	1
$D = 0$ and $S_0, S_1, S_2, S_3 > 0$ without triple root condition	2	1
$D = 0$ and $S_1 \leq S_0 = 0 \leq S_2$ and $S_3 > 0$	0	0
$D = 0$ and $S_1 \leq S_3 = 0 \leq S_2$ and $S_0 > 0$	0	0
$D = 0$ and *else*	2	1

If the discriminant is zero, then in the case of one positive root, the two negative roots come together (one steady state). In the case of discriminant zero and three positive roots, then at least two roots come together (at most two steady states); a triple root occurs if and only if the following *triple root condition* holds:

$$3S_0S_2 \;=\; S_1^2 \quad \text{and} \quad 27S_0^2S_3 \;=\; S_1^3. \tag{4}$$

These equations are precisely what must hold in order for p to have the form $p(x) = -(x-\alpha)^3$. Finally, stability analysis in this one-dimensional system is easy, and this completes the analysis for the networks without boundary steady states. The remaining cases can be classified similarly to complete the entries of Table 1. To parametrize the behavior of the Square, we simply reduce to the case when each of its parameters κ_{12}, κ_{14}, κ_{21}, κ_{23}, κ_{32}, κ_{34}, κ_{41}, and κ_{43} are positive and all others are zero.

By determining which sign vectors in $(0,+)^{12}$ can be realized by a vector of parameters that yields multistationarity, we find a necessary and sufficient condition for a directed graph on the four complexes of the Square to admit multistationarity. This condition is that the graph must include the edges labeled by κ_{23} and κ_{41} and at least one edge directed from the vertex c_1^3 or c_2^3. In this case, for appropriate rate parameters arising from Table 1, the dynamical system has multiple steady states. Therefore, we can enumerate the reversible networks on the four complexes that admit multistationarity: there is one network with all six (bi-directional) edges, four with five edges, six (including the Square) with four edges, four with three edges, and one with two edges. These sixteen networks comprise the family of "smallest" multistationary networks. For the two-edge network, the decomposition from Table 1 is depicted in Figure 1 in the next section.

5 Subnetworks of the Square

Subnetworks of the Square are obtained by removing edges. From the parametrization in the previous section, we know that up to symmetry between c_1 and c_2, only two reversible subnetworks of the Square exhibit multiple steady states.

The first network is obtained by removing the bottom edge of the Square. In other words A_κ is replaced by

$$A_\kappa \quad = \quad \begin{pmatrix} -\kappa_{12}-\kappa_{14} & \kappa_{12} & 0 & \kappa_{14} \\ \kappa_{21} & -\kappa_{21}-\kappa_{23} & \kappa_{23} & 0 \\ 0 & \kappa_{32} & -\kappa_{32} & 0 \\ \kappa_{41} & 0 & 0 & -\kappa_{41} \end{pmatrix}.$$

In this subnetwork, the four parameters of Theorem 1 are the same as those of the Square. The system is a toric dynamical system if and only if the following four binomial generators of M_G vanish:

$$\begin{aligned} \kappa_{14}\kappa_{32} &- \kappa_{23}\kappa_{41}, \\ \kappa_{12}\kappa_{32}\kappa_{41} &- \kappa_{14}\kappa_{21}\kappa_{23}, \\ \kappa_{14}^2\kappa_{21} &- \kappa_{12}\kappa_{41}^2, \\ \kappa_{12}\kappa_{32}^2 &- \kappa_{21}\kappa_{23}^2. \end{aligned}$$

We note that both κ_{23} times the third binomial and κ_{14} times the fourth binomial are in the ideal generated by the first two binomials. Therefore, an assignment of positive parameters for this network defines a toric dynamical system if and only if the following two equations hold: $\kappa_{14}\kappa_{32} = \kappa_{23}\kappa_{41}$ and $\kappa_{12}\kappa_{32}\kappa_{41} = \kappa_{14}\kappa_{21}\kappa_{23}$.

The second subnetwork of the Square is obtained by removing one additional edge, the one between the vertices labeled by c_1^3 and $c_1c_2^2$. The new A_κ is

$$A_\kappa \quad = \quad \begin{pmatrix} -\kappa_{14} & 0 & 0 & \kappa_{14} \\ 0 & -\kappa_{23} & \kappa_{23} & 0 \\ 0 & \kappa_{32} & -\kappa_{32} & 0 \\ \kappa_{41} & 0 & 0 & -\kappa_{41} \end{pmatrix}.$$

The network graph G is now disconnected, and p reduces to

$$p(x) \quad = \quad -\kappa_{14}x^3 + \kappa_{41}x^2 - \kappa_{23}x + \kappa_{32}.$$

The discriminant of p is

$$D \quad = \quad -27\kappa_{14}^2\kappa_{32}^2 - 4\kappa_{14}\kappa_{23}^3 + 18\kappa_{14}\kappa_{23}\kappa_{32}\kappa_{41} + \kappa_{23}^2\kappa_{41}^2 - 4\kappa_{32}\kappa_{41}^3.$$

Further, the toric condition reduces to the single equation

$$\kappa_{23}\kappa_{41} \quad = \quad \kappa_{14}\kappa_{32},$$

which defines the *Segre variety*. A single equation suffices to define the space of toric dynamical systems; this corresponds to the fact that this subnetwork

has deficiency one, while the previous subnetwork has deficiency two. The semi-algebraic decomposition of the previous section for this four-parameter network can be depicted in three dimensions by setting one parameter to be one, in other words, by scaling the equations (1); this is displayed in Figure 1.

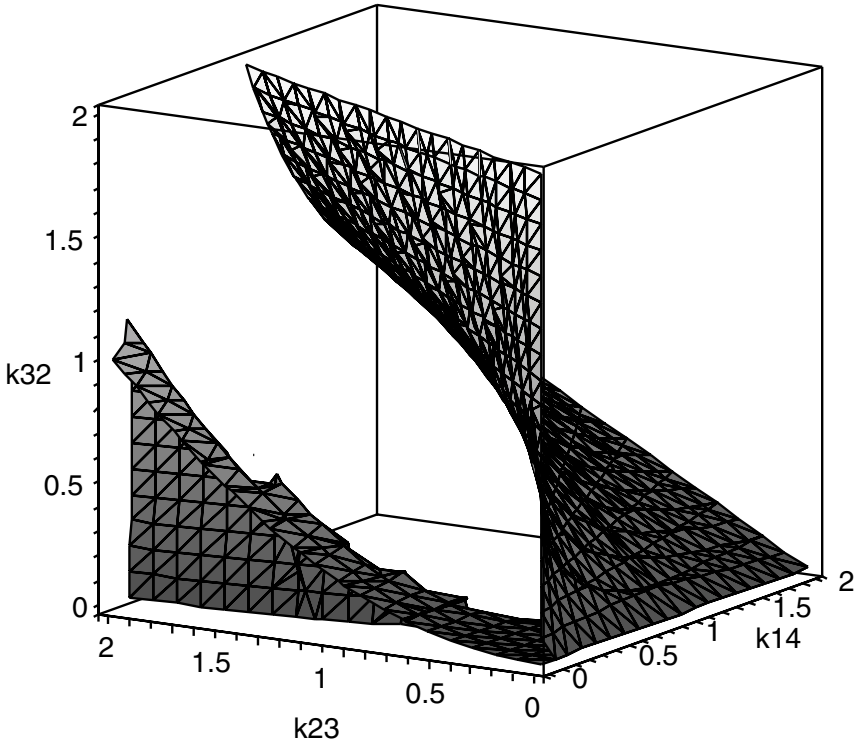


Fig. 1. This depicts the semi-algebraic decomposition of Section 4 for the subnetwork of the Square in which only the vertical edges remain and $\kappa_{41} = 1$. At the left is the discriminant-zero locus. Parameter vectors lying below this surface give rise to dynamical systems with three steady states. Those above the surface yield one steady state; these include parameters of the toric dynamical systems, which are the points on the Segre variety which appears on the right. Parameters on the discriminant-zero locus correspond to systems with either one (if $3\kappa_{14}\kappa_{32} = \kappa_{23}$) or two steady states. This figure was created using `Maple`.

We remark that Horn and Jackson performed the same parametrization for the following special rate constants:

$$(\kappa_{12},\ \kappa_{14},\ \kappa_{21},\ \kappa_{23},\ \kappa_{32},\ \kappa_{34},\ \kappa_{41},\ \kappa_{43}) \quad = \quad (\epsilon,\ 0,\ 1,\ 0,\ \epsilon,\ 0,\ 1,\ 0),$$

where $\epsilon > 0$. Their results are summarized as Table 1 in [16]. Their analysis notes that any instance of three steady states can be lifted to establish the same in the (reversible) Square. In other words, in a small neighborhood in $\mathbb{R}^8_{\geq 0}$ of a vector of parameters that yields three steady states of the directed Square, there is a vector of parameters for the bi-directional Square that also exhibits multistationarity.

The specific criterion for when lifting of this form is possible appears in Theorem 2 of Conradi *et al.* [4]. As this approach is widely applicable, further analysis

of small networks may be fruitful for illuminating the dynamics of larger biochemical networks.

We have seen that the family of Square networks is the smallest class of bistable mass-action kinetics networks. Whether nature has implemented one of these (perhaps with additional components to provide robustness) in a biological setting is as yet unknown, but it is also remarkable that these networks exhibit a simple switch mechanism, which we now explain. Consider the case of three steady states. The corresponding positive roots $x_1 < x_2 < x_3$ of p in Section 4 are the equilibria for the ratio of concentrations c_1/c_2. To switch from the low stable equilibrium x_1 to the high stable equilibrium x_3 is easy: simply increase the concentration ratio c_1/c_2 past x_2, and the dynamics will do the rest.

Acknowledgments. Bernd Sturmfels posed the question of determining the smallest bistable network and provided guidance for this work. We thank Carsten Conradi, Gheorge Craciun, Lior Pachter, and Jörg Stelling for helpful discussions. Anne Shiu was supported by a Lucent Technologies Bell Labs Graduate Research Fellowship.

References

1. Basu, S., Pollack, R., Roy, M.-F.: Algorithms in Real Algebraic Geometry. Springer, Berlin (2006)
2. Cherry, J., Adler, F.: How to make a biological switch. J. Theor. Biol. 203(2), 117–133 (2000)
3. Conradi, C., Flockerzi, D., Raisch, J.: Multistationarity in the activation of a MAPK: parametrizing the relevant region in parameter space. Math. Biosciences 211(1), 105–131 (2008)
4. Conradi, C., Flockerzi, D., Raisch, J., Stelling, J.: Subnetwork analysis reveals dynamic features of complex (bio)chemical networks. P. Natl. Acad. Sci. 104(49), 19175–19180 (2007)
5. Craciun, G., Dickenstein, A., Shiu, A., Sturmfels, B.: Toric dynamical systems (arXiv:0708.3431)
6. Craciun, G., Feinberg, M.: Multiple equilibria in complex chemical reaction networks: I. The injectivity property. SIAM J. Appl. Math. 65(5), 1526–1546 (2005)
7. Craciun, G., Feinberg, M.: Multiple equilibria in complex chemical reaction networks: II. The species-reactions graph. SIAM J. Appl. Math. 66(4), 1321–1338 (2006)
8. Dentin, R., Hedrick, S., Xie, J., Yates, J., Montminy, M.: Hepatic Glucose Sensing via the CREB Coactivator CRTC2. Science 319(5868), 1402–1405 (2008)
9. Ellison, P.: The advanced deficiency algorithm and its applications to mechanism discrimination. PhD Thesis, University of Rochester (1998)
10. Ellison, P., Feinberg, M.: CRNT Toolbox, `http://www.che.eng.ohio-state.edu/~feinberg/crnt/`
11. Feinberg, M.: Chemical oscillations, multiple equilibria, and reaction network structure. In: Stewart, W., Rey, W., Conley, C. (eds.) Dynamics of reactive systems, pp. 59–130. Academic Press, New York (1980)
12. Feinberg, M.: The existence and uniqueness of steady states for a class of chemical reaction networks. Arch. Ration. Mech. Anal. 132, 311–370 (1995)

13. Feinberg, M.: Lectures on chemical reaction networks. Notes of lectures given at the Mathematics Research Center of the University of Wisconsin in 1979 (1979), http://www.che.eng.ohio-state.edu/~feinberg/LecturesOnReactionNetworks
14. Horn, F.: Dynamics of open reaction systems II. Stability and the complex graph. Proc. Royal Soc. A. 334, 313–330 (1973)
15. Horn, F.: Stability and complex balancing in mass-action systems with three complexes. Proc. Royal Soc. A. 334, 331–342 (1973)
16. Horn, F., Jackson, R.: General mass action kinetics. Arch. Rat. Mech. Anal. 47, 81–116 (1972)
17. Laurent, M., Kellershohn, N.: Multistability: a major means of differentiation and evolution in biological systems. Trends Biochem. Sci. 24, 418–422 (1999)
18. Lisman, J.: A Mechanism for Memory Storage Insensitive to Molecular Turnover: A Bistable Autophosphorylating Kinase. Proc. Natll. Acad. Sci. 82(9), 3055–3057 (1985)
19. Segel, L.: Multiple attractors in immunology: theory and experiment. Biophys. Chem. 72(1-2), 223–230 (1998)
20. Sturmfels, B.: Solving systems of polynomial equations. American Mathematical Society, Providence (2002)
21. Wang, D., Xia, B.: Stability analysis of biological systems with real solution classification. In: ISSAC 2005: Proceedings of the 2005 international symposium on Symbolic and algebraic computation, pp. 354–361. ACM, New York (2005)

Local Structure and Behavior of Boolean Bioregulatory Networks

Heike Siebert

DFG Research Center Matheon,
Freie Universität Berlin, Arnimallee 6, D-14195 Berlin, Germany
siebert@mi.fu-berlin.de

Abstract. A well-known discrete approach to modeling biological regulatory networks is the logical framework developed by R. Thomas. The network structure is captured in an interaction graph, which, together with a set of Boolean parameters, gives rise to a state transition graph describing the dynamical behavior. Together with E. H. Snoussi, Thomas later extended the framework by including singular values representing the threshold values of interactions. A systematic approach was taken in [10] to link circuits in the interaction graph with character and number of attractors in the state transition graph by using the information inherent in singular steady states. In this paper, we employ the concept of local interaction graphs to strengthen the results in [10]. Using the local interaction graph of a singular steady state, we are able to construct attractors of the regulatory network from attractors of certain subnetworks. As a comprehensive generalization of the framework introduced in [10], we drop constraints concerning the choice of parameter values to include so-called context sensitive networks.

1 Introduction

In biology, regulatory networks are often visualized as cartoons that illustrate which components of a system interact with each other. A verbal description of the system's behavior clarifies the processes captured in the cartoon. Logical approaches are an intuitive way to model such systems in a mathematical framework. In the 70's, R. Thomas introduced a discrete formalism, which has been continuously further developed and successfully applied to biological problems (see [13], [14] and references therein). Network components are represented by Boolean variables. The structure of the network is captured in a directed, signed graph called interaction graph. Edges represent interactions between components. The sign of an edge signifies whether an activating or inhibiting influence is exerted, provided the tail component of the edge is active, i. e., has value 1. Boolean parameter values specify a function that determines the dynamical behavior. Biologically realistic rules are employed to derive a state transition graph from the Boolean function, which amounts to a non-deterministic representation of all possible behaviors of the system.

This framework has been extended over the years. Network components were allowed to have more than two activity levels, interactions were associated with a

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 185–199, 2008.

threshold value determining when the interaction becomes effective. R. Thomas and E.H. Snoussi used the threshold values, which they called *singular values*, to obtain a better understanding of the system's dynamics. In [11], they focussed on the relation between singular steady states and feedback circuits in the interaction graph of the network. We adapted these ideas to a Boolean setting in [10]. Despite the high level of abstraction, the introduction of singular states proved a useful tool for uncovering relations between structure and dynamics of bioregulatory networks. In this paper, we generalize and develop the results in [10] further. As a first step, we allow characteristics, i. e., the sign of network interactions to depend on the current state of the system. Whether a component has an activating or inhibiting influence on its target may depend on the activity of certain co-factors. A well-known example is the DNA-binding protein TCF which can repress as well as activate the same target genes. TCF acts as activator in the presence of β-catenin, induced by WNT signaling, while the co-expression of the protein TLE converts TCF into a repressor. We call systems including such ambiguous interactions *context sensitive*. Adaptations in the definition of interaction graphs, parameters and singular steady states allow us to include context sensitive systems in our considerations. Furthermore, we exploit the concept of local interactions graphs. It was already successfully used in [6] and [5], and allows for a better understanding of what structures in the interaction graph influence the system's behavior in a given state. This view enables us to focus on the behavior of subnetworks obtained by projection, and from that draw conclusions about the network dynamics.

The organization of the paper is as follows. in Sect. 2 we introduce the Boolean framework used to describe (possibly context sensitive) regulatory networks. We show in Sect. 3 that the set of functions arising from interaction graphs and associated parameter values corresponds to the set of Boolean function $f : \mathcal{B}^n \to \mathcal{B}^n$. We then define the local interaction graph of a given state. Subsequently, we introduce singular steady states. In Sect. 5, we employ the concept of local interaction graphs for singular steady states. The resulting view on the network dynamics allows us to derive certain characteristics of the state transition graph from the behavior of suitable subnetworks. We end the paper with concluding remarks and perspectives for future work.

2 Regulatory Networks

As already mentioned, a directed, signed graph is used in the Thomas formalism to capture the network structure of a regulatory system. We are now interested in a more general representation that allows for the interaction sign to depend on the current state of the system. To accurately describe the structure of such context sensitive networks we use directed multigraphs that allow for parallel edges. Multigraphs have been used in a similar way in [2]. We set $\mathcal{B} := \{0, 1\}$.

Definition 1. *An* interaction (multi-)graph *(or bioregulatory (multi-)graph)* $\mathcal{I}$ *is a labeled directed multigraph with vertex set* $V := \{\alpha_1, \ldots, \alpha_n\}$, $n \in \mathbb{N}$, *and edge set* $E \subseteq V \times V \times \{+, -\}$.

The vertices $\alpha_1, \ldots, \alpha_n$ represent the components of the regulatory network such as genes, RNA, or proteins. We view each component α_i as a variable that adopts values in $\mathcal{B}$. The value 1 signifies that the component is active, i. e., it influences its interaction targets according to the interaction signs. For example, if some substance concentration needs to cross a threshold in order to influence some target component, then the corresponding Boolean value is 0 as long as the concentration is below, and 1 if the concentration is above the threshold.

When analyzing the interaction graph of a network we are interested in certain structural motives. We focus on so-called *(feedback) circuits*. Here, a circuit is a tuple $(e_1, \ldots, e_r)$ of edges $e_i = (k^i, l^i, \varepsilon) \in E$ such that all k^i, $i \in \{1, \ldots, r\}$, are pairwise distinct, and $l^i = k^{i+1}$ for all $i \in \{1, \ldots, r\}$ modulo r. The *sign of a circuit* is the product of the signs of its edges. Note that in a multigraph a circuit is not uniquely determined by its vertices. Figure 1 shows an interaction graph with two circuits consisting of the vertices α_2 and α_3: the positive circuit $((\alpha_2, \alpha_3, +), (\alpha_3, \alpha_2, +))$ and the negative circuit $((\alpha_2, \alpha_3, +), (\alpha_3, \alpha_2, -))$.

To simplify notation, we identify each vertex α_i with its index i, and denote $e_{ij}^{\varepsilon} := (i, j, \varepsilon)$ for all $(i, j, \varepsilon) \in E$. For each α_i we denote by $Pred(\alpha_i)$ the set of *predecessors* of α_i, i. e., the set of vertices α_j such that there is an edge $(\alpha_j, \alpha_i, \varepsilon)$ for some $\varepsilon \in \{+, -\}$ in E. To identify parallel edges we set $E'' = \{(i, j) \mid \exists e_{i,j}^{\varepsilon}, e_{i,j}^{\varepsilon'} \in E : \varepsilon \neq \varepsilon'\}$ and $E' = E \setminus E''$.

An interaction graph holds no information about dynamical behavior. Next we give a formal definition of the term *bioregulatory network* that includes information on structure as well as dynamics. The notation is based on ideas introduced in [1] and [8].

Definition 2. *Let $\mathcal{I} = (V, E)$ be an interaction graph comprising n vertices. A* state *of the system described by $\mathcal{I}$ is a tuple $s \in \mathcal{B}^n$. The set of (regular)* resource edges $R_j(s) = R_j^{\mathcal{I}}(s)$ *of α_j in state $s = (s_1, \ldots, s_n)$ is the set*

$$\{(\alpha_i, \alpha_j, \varepsilon) \in E \mid (\varepsilon = + \wedge s_i = 1) \vee (\varepsilon = - \wedge s_i = 0)\}.$$

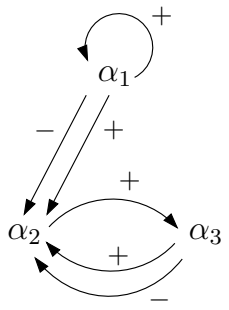
Given a set

$$K(\mathcal{I}) := \{K_{j,R_j(s)} \mid j \in \{1, \ldots, n\},\ s \in \mathcal{B}^n\}$$

of (logical) parameters*, which adopt values in $\mathcal{B}$, we define the Boolean function $f = f^{K(\mathcal{I})} : \mathcal{B}^n \to \mathcal{B}^n$, $s \mapsto (K_{1,R_1(s)}, \ldots, K_{n,R_n(s)})$. The pair $N := (\mathcal{I}, f)$ is called* bioregulatory network*.*

The behavior of a component α_j is determined by the influences its predecessors exert on it. The set of resource edges $R_j(s)$ contains all edges that contribute to an activation of α_j in state s. Note that here the absence of an inhibiting influence (represented by a negative edge) is interpreted as an activating influence on the target component. With this interpretation we have that for every $s \in \mathcal{B}^n$ there is $\varepsilon \in \{+, -\}$ such that $e_{ij}^{\varepsilon} \in R_j(s)$, if $(i, j) \in E''$. If $(i, j) \in E'$, then $R_j(s)$ may or may not contain the corresponding edge $e_{i,j}^{\varepsilon}$, depending on s.

For $j \in \{1, \ldots, n\}$, set $M_j^{\mathcal{I}} := M_j := \{R_j(s) \mid s \in \mathcal{B}^n\}$. Then, by the above considerations, each $M \in M_j$ can be written as $M = \bigcup_{i \in Pred(j)} L_i$ with



$$
\begin{array}{lcl}
K_{1,\emptyset} & = & 0 \\
K_{1,\{e^+_{11}\}} & = & 1 \\
K_{2,\{e^+_{12},e^+_{32}\}} & = & 0 \\
K_{2,\{e^+_{12},e^-_{32}\}} & = & 1 \\
K_{2,\{e^-_{12},e^+_{32}\}} & = & 1 \\
K_{2,\{e^-_{12},e^-_{32}\}} & = & 0 \\
K_{3,\emptyset} & = & 0 \\
K_{3,\{e^+_{23}\}} & = & 1
\end{array}
$$

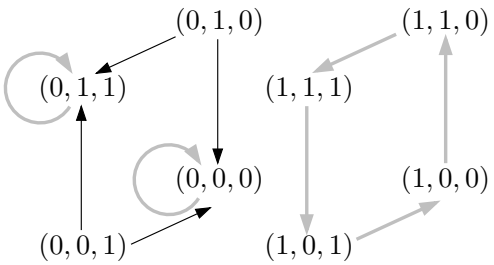


Fig. 1. Interaction graph of a system comprising three components, a list of all parameters with an assignment of Boolean values, and the corresponding state transition graph. The heavier gray edges indicate attractors.

$L_i = \{e^{\varepsilon}_{i,j}\}$ for some $\varepsilon \in \{+,-\}$, if $(i,j) \in E''$, and $L_i = \emptyset$ or $L_i = \{e^{\varepsilon}_{i,j}\} \subset E$, if $(i,j) \in E'$. By definition we have $K(\mathcal{I}) = \{K_{j,M} \mid j \in \{1,\ldots,n\},\, M \in M_j\}$.

The choice of parameter values should be consistent with the information inherent in the interaction graph. We require that each edge represented in the interaction graph should have a notable effect on the system's dynamics. Moreover, the edge's character given by its sign should be reflected in its dynamical impact. To formalize this requirement we again have to distingish between edges in E' and E''. For $e=(i,j,\varepsilon)$, $(i,j) \in E'$, we have $M \cup \{e\} \in M_j$ for all $M \in M_j$, and we demand that $K_{j,M} \leq K_{j,M\cup\{e\}}$ for all $M \in M_j$. Recall that the addition of an edge to the set of resources always signifies increasing activating influence, since the absence of inhibition is interpreted as activating influence. The condition ensures that increasing activating influence does not result in a decrease of component activity level. To ensure that e, at least for some state, has a notable impact on the dynamics, we extend the condition and get:

$$\forall\, M \in M_j \,:\, K_{j,M} \leq K_{j,M\cup\{e\}} \quad \text{and} \quad \exists\, M' \in M_j \,:\, K_{j,M'} < K_{j,M'\cup\{e\}}\,. \quad (1)$$

In the case $(i,j) \in E''$, there exists $e' = (i,j,\varepsilon')$ with $\varepsilon \neq \varepsilon'$. Since α_i influences α_j positively as well as negatively depending on the current state, we cannot impose a general monotonicity condition on the parameters as in the first part of (1). However, again we require that there is at least one state where the addition of e to the set of resources induces an increase in the parameter value. Otherwise the edge e would be superfluous. Since in every given state either e or e' is contained in the set of resources, we compare parameter values for sets $M \in M_j$ and $(M \setminus \{e'\}) \cup \{e\}$. We obtain the condition

$$\exists\, M' \in M_j \,:\, K_{j,M'} < K_{j,(M'\setminus\{e'\})\cup\{e\}}\,. \quad (2)$$

We call edges that satisfy condition (1) resp. (2) *functional*. This concept of functionality is an adaptation of the notion of functionality introduced in [10]. In the following, we always assume that all edges in the interaction graph are *functional*.

In Fig. 1 an interaction graph and a choice of parameter values are given. For α_1 and α_3 the parameters depend on whether or not the single positive edge

ending in α_1 resp. α_3 is effective or ineffective. We have $R_1(s) = \emptyset$ for all states s with $s_1 = 0$, and $R_1(s) = \{e_{11}^+\}$ for all s with $s_1 = 1$. Thus $M_1 = \{\emptyset, \{e_{11}^+\}\}$, and similarly $M_3 = \{\emptyset, \{e_{23}^+\}\}$. The choice of Boolean values for the parameters satisfies condition (1) and ensures the functionality of the edges e_{11}^+ and e_{23}^+. The component α_2 is influenced by both α_1 and α_3 via two parallel edges, respectively. Thus the set of resources is never empty. For example, we have $R_2((0,0,1)) = \{e_{12}^-, e_{32}^+\}$. Overall, we get $M_2 = \{\{e_{12}^+, e_{32}^+\}, \{e_{12}^+, e_{32}^-\}, \{e_{12}^-, e_{32}^+\}, \{e_{12}^-, e_{32}^-\}\}$. Again the choice of parameter values renders all edges functional. A closer look allows the following interpretation. If α_1 has activity level 0, then the influence of α_3 on α_2 corresponds to an activating influence: if α_3 is inactive, α_2 tends to inactivity represented by the parameter $K_{2,\{e_{12}^-, e_{32}^-\}} = 0$, and if α_3 is active α_2 tends to activity since $K_{2,\{e_{12}^-, e_{32}^+\}} = 1$. If α_1 has value 1, then the situation is reversed and α_3 inhibits α_2. The system is context sensitive.

A different choice of parameter values illustrates the concept of functionality. If we set $K_{2,\{e_{12}^+, e_{32}^+\}} = K_{2,\{e_{12}^-, e_{32}^+\}} = 1$ and $K_{2,\{e_{12}^+, e_{32}^-\}} = K_{2,\{e_{12}^-, e_{32}^-\}} = 0$, then verification of conditions (1) and (2) shows that e_{12}^+, e_{12}^- and e_{32}^- are not functional. Only the edge e_{32}^+ is functional and influences the system's dynamics.

In [10], the parameters correspond to sets of resource vertices, i. e., the influence of one component on another cannot change depending on the current state of the system. The network shown in Fig.1 cannot be represented with that restriction. However, the notion of resource edges and resource vertices are equivalent, if there are no parallel edges in the interaction graph.

The parameters determine the behavior of the system as follows. The Boolean value of the parameter $K_{j,R_j(s)}$ indicates how the activity level, i. e., the value of the component α_j will evolve from its value in state s. It will increase (resp. decrease) if the parameter value is greater (resp. smaller) than s_i. The activity level stays the same if both values are equal. Thus, the function f maps a state s to the state the system tends to evolve to. However, if a state and its image differ in more than one component, we take the following consideration into account. In a biological system two different processes of change in activity level represented by the value change of two distinct components will not take the exact same amount of time. Thus we assume that in the discrete dynamical representation a state differs from its successor in at most one component. This procedure is called *asynchronous update* in Thomas' framework. By applying this idea we derive a non-deterministic representation of the dynamics which we again formalize as a directed graph.

Definition 3. *The* state transition graph $\mathcal{S}_N$ *describing the dynamics of the network N is a directed graph with vertex set $\mathcal{B}^n$. For states $s = (s_1, \ldots, s_n)$ and $s' = (s'_1, \ldots, s'_n)$, there is an edge $s \to s'$ if and only if $s' = f(s) = s$ or $s'_i = f_i(s)$ for some $i \in \{1, \ldots, n\}$ satisfying $s_i \neq f_i(s)$ and $s'_j = s_j$ for all $j \neq i$.*

On the right in Fig. 1 we see the state transition graph corresponding to the given interaction graph and parameters. The dynamics are non-deterministic. For example, there are two edges leaving the state (0,1,0), representing two different behaviors of the system.

3 Boolean Functions and Local Interaction Graphs

In the formalism introduced above the dynamical behavior is determined by a Boolean function that is consistent with the underlying interaction graph. In the following we show that for every Boolean function $g : \mathcal{B}^n \to \mathcal{B}^n$ there exists an interaction graph that is consistent with g.

Proposition 1. *Let $g : \mathcal{B}^n \to \mathcal{B}^n$ be a Boolean function. Then there exists an interaction graph $\mathcal{I} = (V, E)$ and a set of parameters $K(\mathcal{I})$ such that $g = f^{K(\mathcal{I})}$.*

Proof. Let $\mathcal{I}^1 = (V, E^1)$ be the interaction graph with $V := \{\alpha_1, \ldots, \alpha_n\}$ and $E^1 := V \times V \times \{+, -\}$, i. e., $\mathcal{I}^1$ includes every possible edge. We set $K^1_{i,R^1_i(s)} := g_i(s)$ for all $i \in \{1, \ldots, n\}$ and $s \in \mathcal{B}^n$, with $R^1_i(s) := R^{\mathcal{I}^1}_i$.

Now, we have to consider the functionality of the edges in $\mathcal{I}^1$. This can be done componentwise via conditions (1) and (2) given in Sect. 2. We then eliminate edges that are not functional and derive a new interaction graph and corresponding parameter values in an iterative procedure, starting with $\mathcal{I}^1$ and the parameter values given above. In the k-th step, if edge $e = (i, j, \varepsilon)$ is functional, we make no alterations on the interaction graph and parameters. If e is not functional, we define a new interaction graph $\mathcal{I}^{k+1} = (V, E^{k+1})$ with $E^{k+1} := E^k \setminus \{e\}$. Clearly, we have $M^{k+1}_j = \{M \setminus \{e\} \mid M \in M^k_j\}$, since e is eliminated in all sets of resources. We then set $K^{k+1}_{j,M\setminus\{e\}} := K^k_{j,M}$ for all $M \in M^k_j$, and keep all other parameters the same as before. By definition the function $f^{K(\mathcal{I}^{k+1})}$ derived from the new parameters still coincides with g.

However, we have to make sure that the order of edges we use to check functionality does not influence the result. That is, we want to show that an edge e is functional in $\mathcal{I}^k$ if and only if it is functional in $\mathcal{I}^{k+1}$, w. r. t. the parameters as chosen above. This follows directly from the relation between parameter values in the k-th and $(k+1)$-th step. For example, let e be a functional edge in $\mathcal{I}^k$, and assume that its parallel edge e' is not functional and deleted in the step resulting in $\mathcal{I}^{k+1}$. To check whether e is still functional we now have to consider condition (1) instead of (2). Let us assume the first part of (1) does not hold. Then there exists $M \in M^{k+1}_j$ such that $K^{k+1}_{j,M} > K^{k+1}_{j,M\cup\{e\}}$. It follows that $e \notin M$. Since we deleted the parallel edge e', we know $e' \notin M$, $M \cup \{e\}, M \cup \{e'\} \in M^k_j$ and $K^k_{j,M\cup\{e'\}} = K^{k+1}_{j,M}$. We set $M' := M \cup \{e\}$. Then we get $K^k_{j,(M'\setminus\{e\})\cup\{e'\}} = K^{k+1}_{j,M} > K^{k+1}_{j,M\cup\{e\}} = K^k_{j,M\cup\{e\}} = K^k_{j,M'}$. This contradicts the assumption that (2) does not hold for e' in $\mathcal{I}^k$. Thus e satisfies the first part of (1). All other functionality statements can be shown with similar methods.

We repeat the iterative procedure for every edge in $\mathcal{I}^1$. After $2n^2$ steps we obtain an interaction graph and parameters consistent with g. □

We persistently emphasize the point that we only deal with functional edges, i. e. edges that have an impact on the dynamics. However, this influence does not have to be effective in the whole state space $\mathcal{B}^n$. If we want to understand the way the structure and dynamics of a system relate to each other, then it is useful

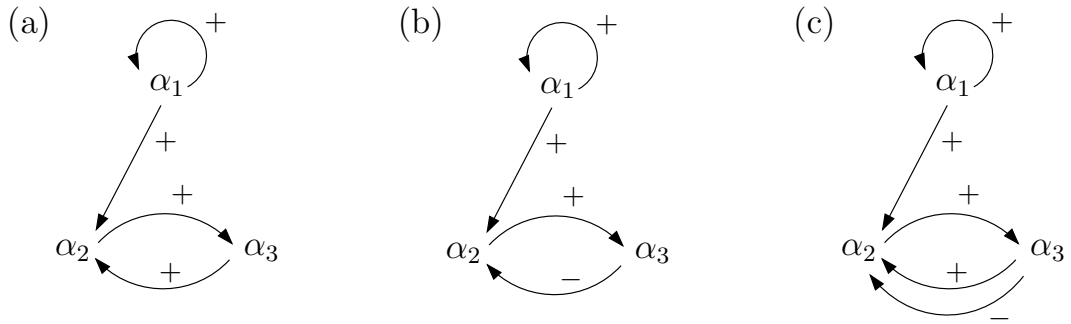


Fig. 2. Local interaction graphs corresponding to the graph and parameters given in Fig. 1. $\mathcal{I}((0,0,0))$ in (a), $\mathcal{I}((1,0,0))$ in (b), $\mathcal{I}((\theta,0,0))$ in (c).

to have a closer look on the effective interactions depending on the current state of the system. To capture those local structural aspects we introduce the concept of *local interaction graphs*. It has already been used in [6] and [5] (see also references therein). In the following, we denote with $\overline{s}^i$ the state that coincides with s in all components $j \neq i$ and takes the value $1 - s_i$ in the i-th component.

Definition 4. *Let $\mathcal{I} = (V, E)$ be an interaction graph with parameter set $K(\mathcal{I})$. Let $s = (s_1, \ldots, s_n) \in \mathcal{B}^n$. Then we denote by $\mathcal{I}(s)$ the graph with vertex set V and edge set $E(s) \subseteq E$. An edge (i, j, ε) is in $E(s)$ if and only if*

$$K_{j,R_j(s)} \neq K_{j,R_j(\overline{s}^i)} \quad \wedge \quad \varepsilon = + \Leftrightarrow s_i = K_{j,R_j(s)} \,.$$

We call $\mathcal{I}(s)$ the (local) interaction graph *in state s.*

Clearly, every edge in a local interaction graph $\mathcal{I}(s)$ is also contained in $\mathcal{I}$, since we use the parameters of the interaction graph $\mathcal{I}$ to characterize the edges in a local interaction graph. More precisely, $\mathcal{I}$ is the union of all graphs $\mathcal{I}(s)$, $s \in \mathcal{B}^n$. We call $\mathcal{I}$ also the *global interaction graph*. Note that there are no parallel edges in a local interaction graph. Figure 2(a) and (b) show the graphs $\mathcal{I}((0,0,0))$ and $\mathcal{I}((1,0,0))$ corresponding to the example given in Fig. 1. The local interaction graphs give us a finer understanding of the way the network components interact. They can be seen as a visualization of the discrete Jacobian matrix of the Boolean function $f^{\mathcal{I}}$ as introduced in [9].

4 Singular States

In our formalism we only consider whether a component is active or not. We now incorporate a threshold value that allows us to express uncertainty in the sense that we do not know if a certain interaction is effective. We already used this concept in [10] for networks without context sensitivity. Again, we mainly use notation introduced in [8]. Throughout this section let $N := (\mathcal{I}, f = f^{K(\mathcal{I})})$ be a bioregulatory network comprising n components.

Definition 5. *Set $\mathcal{B}_\theta := \{0, \theta, 1\}$, where θ is a symbolic representation of the threshold value and satisfies the order $0 < \theta < 1$. We allow each regulatory component α_i to take values in $\mathcal{B}_\theta$. The values 0 and 1 are called* regular values *and*

θ is called singular value*. The elements of $\mathcal{B}_\theta^n$ are called* states*. If all components of a state are regular, it is called* regular state*, else it is called* singular state*. For every state $s = (s_1, \ldots, s_n)$ we define $J(s) := \{i \in \{1, \ldots, n\} \mid s_i = \theta\}$.*

We call $|a, b|$ a *qualitative value* if $a, b \in \mathcal{B}$ and $a \leq b$. The qualitative value $|0, 0|$ is identified with the regular value 0, $|1, 1|$ with the regular value 1, and $|0, 1|$ with the singular value θ. The relations $<$, $>$, and $=$ are used with respect to this identification.

Definition 6. *We define for all $i \in \{1, \ldots, n\}$*

$$f^\theta = f^{K(\mathcal{I}),\theta} : \mathcal{B}_\theta^n \to \mathcal{B}_\theta^n \quad \text{by} \quad f_i^\theta(s) = |K_{i,\min(s)}, K_{i,\max(s)}|\,,$$

where $K_{i,\min(s)} := \min\{\, K_{i,R_i(s')} \mid s' \in \mathcal{B}^n\,,\ s'_j = s_j \ \text{for all}\ j \notin J(s)\}$ and $K_{i,\max(s)} := \max\{\, K_{i,R_i(s')} \mid s' \in \mathcal{B}^n\,,\ s'_j = s_j \ \text{for all}\ j \notin J(s)\}$. We call $s \in \mathcal{B}_\theta^n$ a steady state if $f^\theta(s) = s$.

The definition of $K_{i,\min(s)}$ and $K_{i,\max(s)}$ ensures that the image of a regular state under f^θ is again a regular state. More specific, we have $f^\theta|_{\mathcal{B}^n} = f$. If a state has singular components, then $K_{i,\min(s)}$ and $K_{i,\max(s)}$ reflect the dynamical behavior of the component i in the two extreme cases that either all singular predecessors of α_i have no activating influence on α_i or they all contribute to an activation of α_i.

Thomas and Snoussi already link singular states to circuits in the interaction graph, albeit in a different framework (see [11]). We have adapted their ideas to a Boolean framework without context sensitivity in [10].

Definition 7. *Let $C = (\alpha_{i_1}, \ldots, \alpha_{i_r})$ be a circuit in $\mathcal{I}$. A state $s = (s_1, \ldots, s_n) \in \mathcal{B}_\theta^n$ is called* characteristic state *of C if $s_{i_l} = \theta$ for all $l \in \{1, \ldots, r\}$.*

In general, a characteristic state of a circuit is not unique. The state $(\theta, \ldots, \theta)$ is characteristic for every circuit in $\mathcal{I}$. A simple modification of the reasoning in [10] leads to the following statement.

Theorem 1. *Every singular steady state is characteristic of some circuit in $\mathcal{I}$.*

A singular steady state s can be characterized using only regular states and the function f. The idea is to check componentwise the behavior for regular states s^+ and s^- that satisfy $K_{i,R_i(s^+)} = K_{i,\max(s)}$ and $K_{i,R_i(s^-)} = K_{i,\min(s)}$ for some $i \in \{1, \ldots, n\}$. The proofs for networks that are not context sensitive are given in [10] and can be easily adapted.

5 Attractors and Local Interaction Graphs of Singular Steady States

In this section we link structural properties of (local) interaction graphs with the dynamical behavior of the system by considering singular steady states. Every possible behavior of the system is captured in the corresponding state transition

graph introduced in Sect. 2. In the following let $N := (\mathcal{I} = (V, E), f = f^{K(\mathcal{I})})$ be a bioregulatory network comprising n components and $\mathcal{S}_N$ the corresponding state transition graph. In addition to standard terminology from graph theory such as paths and cycles we us the following concepts.

Definition 8. *An infinite path* $(s_0, s_1, \dots)$ *in* $\mathcal{S}_N$ *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* $s^1, s^2 \in A$ *there is a path from* s^1 *to* s^2 *in* $\mathcal{S}_N$. *A cycle* $C := (s^1, \dots, s^r, s^1)$, $r \geq 2$, *is called a* trap cycle *if every* s^j, $j \in \{1, \dots, r\}$, *has only one outgoing edge in* $\mathcal{S}_N$, *i. e., the trajectory starting in* s^1 *is unique.*

In other words, the attractors correspond to the terminal strongly connected components of the graph. Regular steady states as well as trap cycles are attractors. The attractors in the state transition graph given in Fig. 1 are the sets containing the steady states, i. e., $\{(0, 0, 0)\}$ and $\{(0, 1, 1)\}$, and the set containing the states of the trap cycle in the graph, i. e., $\{(1, 0, 0), (1, 1, 0), (1, 1, 1), (1, 0, 0)\}$.

The behavior of a system becomes, at least to some degree, predictable and stable inside an attractor. Often, a sensible biological interpretation can be found for an attractor. In cell differentiation, the different stable states reached at the end of development may be represented by distinct steady states in the state transition graph. Attractors of cardinality greater than one imply cyclic behavior, and thus can often be identified with homeostasis of sustained oscillatory activity, as can be found in the cell cycle or circadian rhythm.

State transition graphs always contain at least one attractor. The proof of the following more precise statement can be found in [10].

Proposition 2. *For every state* $s \in \mathcal{B}^n$ *exists a trajectory in* $\mathcal{S}_N$ *which starts in* s *and leads to an attractor.*

If some vertex α_i in $\mathcal{I}$ does not have a predecessor, then clearly $a_i = K_{i,\emptyset}$ for every state $a = (a_1, \dots, a_n)$ in an attractor. Similarly, we know the values a_j for vertices the only predecessor of which is α_i, and so on. Throughout this section we assume that every vertex in $\mathcal{I}$ has a predecessor. Note that an input value in the sense of a component that maintains its current activity level independent of the values of the other components is represented as a vertex with its only incoming edge being a positive selfloop.

In the following we have a closer look at the information concerning the network dynamics in general and the attractors in particular that is inherent in the existence and properties of singular steady states. We also want to exploit structural information. As a first step, we adapt the concept of local interaction graphs to singular states. Recall that $J(s)$ is the set of all singular components of a state $s \in \mathcal{B}^n_\theta$.

Definition 9. *Let* $s = (s_1, \dots, s_n) \in \mathcal{B}^n_\theta$. *We denote by* $\mathcal{I}(s)$ *the (multi-)graph with vertex set* V *and edge set* $E(s)$. *An edge* e *is in* $E(s)$ *if and only if there exists a regular state* $s' = (s'_1, \dots, s'_n)$ *such that* $s'_i = s_i$ *for all* $i \notin J(s)$ *and* $e \in E(s')$, *where* $E(s')$ *denotes the edge set of the interaction graph* $\mathcal{I}(s')$ *in* s'. *Again, we call* $\mathcal{I}(s)$ *the (local) interaction graph in* s.

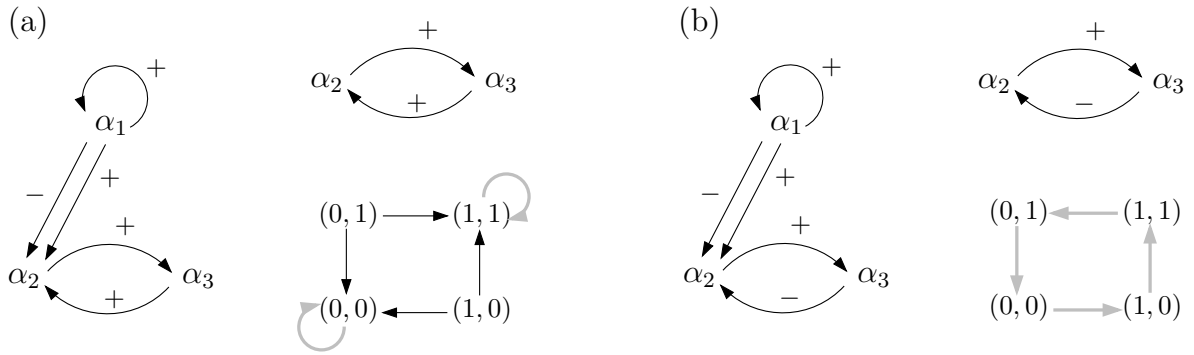


Fig. 3. Local interaction graph $\mathcal{I}((0,\theta,\theta))$ on the left, $\mathcal{I}^\theta((0,\theta,\theta))$ and corresponding state transition graph on the right of (a). Local interaction graph $\mathcal{I}((1,\theta,\theta))$ on the left, $\mathcal{I}^\theta((1,\theta,\theta))$ and corresponding state transition graph on the right of (b). Attractors are indicated by heavier gray edges.

Note that the interaction graph in a singular state may have parallel edges. In Fig. 2 (c) we see the local interaction graph in state $(\theta,0,0)$, which is the union of the graphs $\mathcal{I}((0,0,0))$ and $\mathcal{I}((1,0,0))$ given in (a) and (b).

A singular steady state s yields stability in the dynamical behavior for the components that do not belong to $J(s)$. To make a more precise statement we introduce notation for a specific subgraph of $\mathcal{I}(s)$. By $\mathcal{I}^\theta(s)$ we denote the (multi-)graph with vertex set $V^\theta(s) := J(s)$ and edge set $E^\theta(s) := \{(i,j,\varepsilon) \in E(s) \mid i,j \in J(s)\}$. That is, we only keep the singular components and interactions between them. We call a graph Z component of $\mathcal{I}^\theta(s)$, if $Z = (V_Z, E_Z)$ is a maximal subgraph of $\mathcal{I}^\theta(s)$ such that for every $k, k' \in V_Z$ exist vertices $k_1, \ldots, k_r \in V_Z$ with $k_1 = k$, $k_r = k'$, and $(k_i, k_{i+1}, \varepsilon) \in E^\theta(s)$ or $(k_{i+1}, k_i, \varepsilon) \in E^\theta(s)$ for some $\varepsilon \in \{+,-\}$ and all $i \in \{1,\ldots,r-1\}$. In Fig. 3 we see for our running example introduced in Fig. 1 the graphs $\mathcal{I}((0,\theta,\theta))$ and $\mathcal{I}^\theta((0,\theta,\theta))$ in (a), as well as the graphs $\mathcal{I}((1,\theta,\theta))$ and $\mathcal{I}^\theta((1,\theta,\theta))$ in (b). Lastly, let C be a circuit in $\mathcal{I}(s)$ such that all edges of C are in $\mathcal{I}^\theta(s)$. Then there exists a component of $\mathcal{I}^\theta(s)$ that contains C. We denote this component by $J_C(s)$. The next lemma shows that the stability of the regular components of a singular steady state is not influenced by value changes in a component Z of $\mathcal{I}^\theta(s)$. Moreover, if $\mathcal{I}^\theta(s)$ has more than one component, the component dynamics are independent of each other. This property is crucial for the remaining results in this section. The proof of the lemma is an adaptation of a similar, less general statement in [10]. Note that in [10] a different definition of $\mathcal{I}^\theta(s)$ is used that does not take the effectiveness of interactions in state s into account.

Lemma 1. *Let $s = (s_1,\ldots,s_n)$ be a singular steady state, and let $Z_1,\ldots,Z_m$ be the components of $\mathcal{I}^\theta(s)$. Consider a union Z of arbitrary components Z_j. Let $\tilde{s} = (\tilde{s}_1,\ldots,\tilde{s}_n) \in \mathcal{B}_\theta^n$ such that $\tilde{s}_i = s_i$ for all $i \notin Z$. Then $f_i^\theta(\tilde{s}) = f_i^\theta(s) = s_i = \tilde{s}_i$ for all $i \notin Z$.*

Proof. First, let us consider $i \in J(s) \setminus Z$. Then $s_j = \tilde{s}_j$ for every $j \in Pred(\alpha_i)$, since there are no predecessors of α_i in Z. Therefore, the sets of resource edges of α_i are not influenced by value changes in Z, i. e., $\{R_i(s') \mid s' \in \mathcal{B}^n$,

$s'_j = s_j$ for all $j \notin J(s)\} = \{R_i(s') \mid s' \in \mathcal{B}^n,\ s'_j = \tilde{s}_j$ for all $j \notin J(\tilde{s})\}$. Then $K_{i,\min(s)} = K_{i,\min(\tilde{s})}$ and $K_{i,\max(s)} = K_{i,\max(\tilde{s})}$, and $f_i^\theta(\tilde{s}) = f_i^\theta(s) = s_i = \tilde{s}_i$.

Now, let $i \notin J(s)$. Since $s_j = \theta$ for all $j \in Z$, we have $J(\tilde{s}) \subseteq J(s)$. Therefore, $\{s' \in \mathcal{B}^n \mid s'_j = \tilde{s}_j$ for all $j \notin J(\tilde{s})\} \subseteq \{s' \in \mathcal{B}^n \mid s'_j = s_j$ for all $j \notin J(s)\}$. It follows that $K_{i,\min(s)} \leq K_{i,\min(\tilde{s})} \leq K_{i,\max(\tilde{s})} \leq K_{i,\max(s)}$. Since $f_i^\theta(s) = s_i$ is regular, we know $K_{i,\min(s)} = K_{i,\max(s)} = s_i$. Thus, $K_{i,\min(\tilde{s})} = K_{i,\max(\tilde{s})} = s_i$ and $f_i^\theta(\tilde{s}) = s_i = \tilde{s}_i$. □

The above lemma shows that we can construct attractors of the state transition graph $\mathcal{S}_N$ from attractors of the dynamics restricted to the components of $\mathcal{I}^\theta(s)$. To give a clear understanding of this construction we need the following notation.

Let s be a singular steady state and Z a component of $\mathcal{I}^\theta(s)$ with $k := \operatorname{card} V_Z$. We may assume that $V_Z = \{\alpha_{l+1}, \ldots, \alpha_{l+k}\}$ for some $l \in \{0, \ldots, n-1\}$. Then Z is an interaction graph comprising k vertices. Now, we want to define the dynamics of Z as the projection of the dynamics of $\mathcal{I}$ with respect to s. We define a parameter set $K(Z)$ according to Def. 2 as the set of all parameters $K^Z_{i,R_i^Z(z)} := K_{i,R_i(\tilde{s})}$ for $z \in \mathcal{B}^k$ and $\tilde{s} \in \mathcal{B}^n$ with $\tilde{s}_i = s_i$ for all $i \notin J(s)$ and $\tilde{s}_i = z_{i-l}$ for all $i \in Z$. The parameters are well defined since there are no predecessors of vertices in Z in $J(s) \setminus Z$. We set $f^{K(Z)} = f^Z : \mathcal{B}^k \to \mathcal{B}^k$, $z \mapsto (K^Z_{1,R_1^Z(z)}, \ldots, K^Z_{k,R_k^Z(z)})$. We then have $f^Z = \pi^Z \circ f^\theta \circ \rho^Z$, where $\rho^Z : \mathcal{B}^k \to \mathcal{B}^n$ with $\rho_i^Z(z) = s_i$ for $i \notin Z$ and $\rho_i^Z(z) = z_{i-l}$ for $i \in Z$, and $\pi^Z : \mathcal{B}^n \to \mathcal{B}^k$ is the projection on the components of Z. Note that f^Z yields always regular values, since the singular values in $J(s) \setminus Z$ do not influence the components in Z. The definitions of parameters and $\mathcal{I}^\theta(s)$ ensure that all edges in Z are functional. According to Prop. 2 the graph $\mathcal{S}_{N^Z}$ contains an attractor. This fact leads to the next theorem.

Theorem 2. *Let $s = (s_1, \ldots, s_n)$ be a singular steady state, and $Z_1, \ldots, Z_m$ be the components of $\mathcal{I}^\theta(s)$. For all $j \in \{1, \ldots, m\}$ let A_j be an attractor of the state transition graph corresponding to the network N^{Z_j} as defined above. Then there exists an attractor A in the state transition graph $\mathcal{S}_N$ such that $a_i = s_i$ for all $a = (a_1, \ldots, a_n) \in A$, $i \notin J(s)$, and $\pi^{Z_j}(A) = A_j$ for all $j \in \{1, \ldots, m\}$.*

Proof. Without loss of generality we may assume that Z_1 contains the vertices $\alpha_1, \ldots, \alpha_{\operatorname{card} Z_1}$, Z_2 contains the vertices $\alpha_{\operatorname{card} Z_1+1}, \ldots, \alpha_{\operatorname{card} Z_1+\operatorname{card} Z_2}$, etc. We set $k := 1 + \sum_{i+1}^m \operatorname{card} Z_i$ and $A := A_1 \times \cdots \times A_m \times \{(s_k, \ldots, s_n)\}$.

First, we show that A is a trap set, i. e., every successor of a state in A is again in A. Let $x \in A$ and x' be a successor of x in $\mathcal{S}_N$. Assume $x \neq x'$. Then there exists $i \in \{1, \ldots, n\}$ such that $x'_i = f^i(x) \neq x_i$ and $x'_j = x_j$ for all $j \neq i$. Lemma 1 yields that $f_j(x) = x_j = s_j$ for all $j \in \{k, \ldots, n\}$. Thus, we find $l \in \{1, \ldots, m\}$ such that $i \in Z_l$. Now, we only have to show that $\pi^{Z_l}(x') \in A^l$. Per definition we have $f^{Z_l}(\pi^{Z_l}(x)) = \pi^{Z_l} \circ f^\theta \circ \rho^{Z_l}(\pi^{Z_l}(x))$. Lemma 1 allows us to ignore the values of components in $J(s) \setminus Z_l$ and we obtain $\pi^{Z_l} \circ f^\theta \circ \rho^{Z_l}(\pi^{Z_l}(x)) = \pi^{Z_l}(f^\theta(x)) = \pi^{Z_l}(f(x))$. Since $i \in Z_l$, we then have $f^{Z_l}_{i-\operatorname{card} Z_{l-1}}(\pi^{Z_l}(x)) = f_i(x) = x'_i \neq x_i = \pi^{Z_l}_{i-\operatorname{card} Z_{l-1}}(x)$, where we set $\operatorname{card} Z_0 = 0$. Per definition there is an edge between $\pi^{Z_l}(x)$ and $\pi^{Z_l}(x')$ in $\mathcal{S}_{N^{Z_l}}$ and, since A_l is an attractor, we have $\pi^{Z_l}(x') \in A_l$.

Now, we have to show that there is a path from x to x' in $\mathcal{S}_N$ for all distinct $x, x' \in A$. First, we prove that if there is an edge from state z to state z', $z \neq z'$, in $\mathcal{S}_N z_l$, $l \in \{1, \ldots, m\}$, then there is an edge from x to x' in $\mathcal{S}_N$ for all states $x, x' \in A$ satisfying $\pi^{Z_l}(x) = z$, $\pi^{Z_l}(x') = z'$, and $x_j = x'_j$ for all $j \notin Z_l$. According to the definition there is $i \in Z$ such that $z_p \neq z'_p = f_p^{Z_l}(z)$ with $p = i - \operatorname{card} Z_{l-1}$. With Lemma 1 follows that $z_p \neq f_p^{Z_l}(z) = \pi_p^{Z_l} \circ f^\theta \circ \rho^{Z_l}(z) = \pi_p^{Z_l}(f(x)) = f_i(x)$ for all $x \in A$ with $x_j = z_j$ for all $j \in Z_l$. For every such x the state x' satisfying $x'_j = x_j$ for all $j \neq i$ and $x'_i = f_i(x) \neq z_i = x_i$ is also in A, and there is an edge from x to x' in $\mathcal{S}_N$.

Let $x, x' \in A$. We set $x_i^1 := x_i$ for all $i \notin Z_1$ and $x_i^1 := x'_i$ for all $i \in Z_1$. For $l \in \{2, \ldots, m\}$ we set $x_i^l := x_i^{l-1}$ for all $i \notin Z_l$ and $x_i^l := x'_i$ for all $i \in Z_l$. Then there exists a path in $\mathcal{S}_N z_1$ from $\pi^{Z_1}(x)$ to $\pi^{Z^1}(x^1)$, since A_1 is an attractor. As seen above, we then can find a path γ_1 from x to x^1 in $\mathcal{S}_N$ such that $\tilde{x}_j = x_j$ for every state $\tilde{x} \in \gamma_1$ and every $j \notin Z_1$. In the same fashion we find a path γ_2 from x^1 to x^2 in $\mathcal{S}_n$ such that $\tilde{x}_j = x_j^1$ for all $\tilde{x} \in \gamma_2$ and $j \notin Z_2$. We continue the procedure for $Z_3, \ldots, Z_m$. Since $x^m = x'$ per definition, combining the paths γ_i in the order of their indices yields a path from x to x' in $\mathcal{S}_N$. □

We illustrate the theorem by considering our running example in Fig. 1. As shown in Fig. 3 (a), the graph $\mathcal{I}^\theta((0, \theta, \theta))$ has only one component Z consisting of a positive circuit containing α_2 and α_3. We derive the parameters $K(Z)$ from those given in Fig. 1 for the global interaction graph. Since $s_1 = 0$, we obtain, according to the above definition, the parameters $K^Z_{2,\emptyset} := K_{2,\{e^-_{12}, e^-_{32}\}} = 0$ and $K^Z_{2,\{e^+_{32}\}} := K_{2,\{e^-_{12}, e^+_{32}\}} = 1$. The parameters for α_3 stay the same, i. e., $K^Z_3, \omega = K_{3,\omega}$ for $\omega \in \{\emptyset, \{e^+_{23}\}\}$. The resulting state transition graph $\mathcal{S}^Z_N$ is also given in Fig. 3 (a) and contains the attractors $\{(0, 0)\}$ and $\{(1, 1)\}$. It follows from Theorem 2 that the sets $\{(0, 0, 0)\}$ and $\{(0, 1, 1)\}$ are attractors in $\mathcal{S}_N$. Similarly, we derive a state transition graph from $\mathcal{I}^\theta((1, \theta, \theta))$ which consists of a negative circuit. The state transition graph is shown in Fig. 3 (b) and contains only one attractor, the set $\{(0, 0), (1, 0), (1, 1), (0, 1)\}$, which has cardinality greater than one. Thus, we find an attractor $\{(1, 0, 0), (1, 1, 0), (1, 1, 1), (1, 0, 1)\}$ in $\mathcal{S}_N$. The state transition graph $\mathcal{S}_N$ is given in Fig. 1 with the attractors emphasized.

In [4] it is shown that isolated circuits always display a characteristic behavior depending on their sign. A positive circuit gives rise to two attractors, more precisely two steady states, a negative circuit results in a cyclic attractor, i. e., an attractor with cardinality greater than one. The situation is much more difficult to analyze if there are many circuits in $\mathcal{I}$, possibly even intertwined. Thomas conjectured in 1981 that the existence of a positive resp. negative circuit in the interaction graph is a necessary condition for the existence of two attractors resp. a cyclic attractor in the state transition graph. The conjectures haven been proven in different settings (see e. g. [12], [5] and [7]). For regulatory networks without context sensitivity, we formulated in [10] a sufficient condition for circuits to display their characteristic behavior using singular steady states. The proof in [10] can be easily adapted to show the next statement.

Lemma 2. *Let I be an interaction graph that contains only one circuit C. If C is a positive circuit, then f has two fixed points. If C is negative, then there exists an attractor with cardinality greater than one in the state transition graph.*

We make some short remarks on the proof. Recall our assumption that every vertex in $\mathcal{I}$ has a predecessor. Since every edge is functional, the state $(\theta, \ldots, \theta)$ is steady. In [10], it is shown that $\mathcal{I}$ then has a particular structure. It consists of the circuit C with possibly directed trees coming out of vertices of C. Those trees may also be interconnected. This structure allows us to explicitly specify values for the vertices of C that remain fix under f^θ in the case of C being positive, or behave like a trap cycle, if C is negative. From this core behavior we can then infer the behavior of the whole graph. Here, we also have to consider that there may be parallel edges outside the circuit C. However, the proof method is still valid. The necessary technical adaptations to the proofs in [10] correspond to those made in the proof of Lemma 1.

The above lemma together with Theorem 2 leads to the following theorem.

Theorem 3. *Let C be a circuit in $\mathcal{I}$ and s a singular steady state characteristic of C. Assume that C is the only circuit in the component $J_C(s)$ of $\mathcal{I}^\theta(s)$. If C is a positive circuit, then f^θ has at least three fixed points and $\mathcal{S}_N$ contains at least two attractors. If C is negative, there is an attractor in $\mathcal{S}_N$ with cardinality greater than one.*

Proof. We may assume that $J_C(s)$ comprises the vertices $\alpha_1, \ldots, \alpha_r$ for some $r \in \{1, \ldots, N\}$. Let at first C be positive. Then $f^{J_C(s)}$ has two fixed points $x, x' \in \mathcal{B}^r$ according to Lemma 2. We define states s^1 and s^2 in $\mathcal{B}^n_\theta$ by $s^1_i := s^2_i := s_i$ for all $i \notin J_C(s)$, $s^1_i := x_i$ and $s^2_i := x'_i$ for all $i \in \{1, \ldots, r\}$. From Lemma 1 follows that the states s^1 and s^2 are steady states. Thus f^θ has three fixed points, since s is distinct from s^1 and s^2. According to Theorem 2 we find attractors A_1 and A_2 in $\mathcal{S}_N$ such that $\pi^{J_C(s)}(A_1) = \{s^1\}$ and $\pi^{J_C(s)}(A_2) = \{s^2\}$.

If C is negative, we find an attractor A' in the state transition graph of the component graph $J_C(s)$ with $\operatorname{card} A' > 1$. Theorem 2 yields an attractor A in $\mathcal{S}_N$ with $\pi^{J_C(s)}(A) = A'$. Thus cardinality of A is also greater than one. □

Theorem 3 is a stronger result than the one obtained in [10], even for networks without context sensitivity. The use of local interaction graphs allows for a more refined picture of the dynamics possible in restricted parts of the state space.

Our running example from Fig. 1 together with Fig. 3 illustrates the theorem. Figure 4 shows that the statement does not hold, if the circuit C is not the only circuit in $J_C(s)$. The state $(\theta, 0, \theta)$ is steady for the bioregulatory network derived from the interaction graph in (a) and the parameters specified in the caption. There are four circuits in $\mathcal{I}^\theta((\theta, 0, \theta))$, two negative and two positive circuits. However, the state transition graph contains only one attractor, namely the set $\{(0, 0, 1)\}$, as is shown in (c). Neither the behavior characteristic for positive circuits nor that characteristic for negative circuits is displayed. Further examples can be found in [10]. However, a system may display the behavior characteristic for a circuit of a given sign, although there is no singular steady

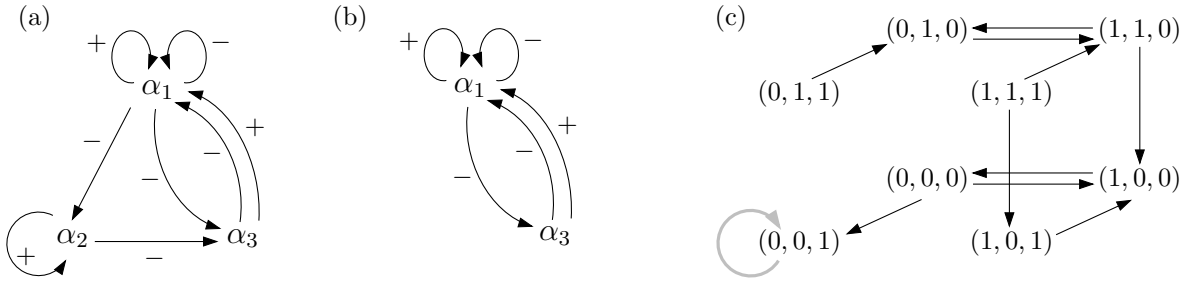


Fig. 4. We choose the parameters for the interaction graph in (a) as $K_{1,\{e_{11}^+,e_{31}^+\}} = K_{1,\{e_{11}^-,e_{31}^-\}} = K_{2,\{e_{12}^-,e_{22}^+\}} = K_{3,\{e_{13}^-,e_{23}^-\}} = 1$ and set all other parameters 0. In (b) the graph $\mathcal{I}^\theta(s)$ for the singular steady state $s = (\theta, 0, \theta)$. In (c) the corresponding state transition graph.

s such that the circuit is the only one in the corresponding component of $\mathcal{I}^\theta(s)$. The condition is not necessary, as illustrated by an example given in [10], Fig. 4.

6 Conclusion

In [10] we started a systematic investigation of the relation between singular steady states and attractors in the state transition graph of regulatory networks, which are described by an interaction graph and Boolean parameters. Among other results, we found sufficient conditions concerning singular steady states and circuits in the interaction graph ensuring the existence of two distinct attractors resp. a cyclic attractor. In this paper, we considerably refine and generalize the results in [10]. We are now able to deal with systems that display context sensitivity resulting in interaction graphs with parallel edges. In Sect. 3 we have shown that in this framework the set of functions arising from interaction graphs and associated parameter values corresponds to the set of Boolean functions $f : \mathcal{B}^n \to \mathcal{B}^n$. To obtain a better understanding of the relation between the structure and the behavior of the system, we employ local interaction graphs, which consist of the interactions influencing the behavior of the system in a given state. Using the local interaction graph of a singular steady state, we are able to construct attractors of the given regulatory network from attractors of subsystems of the network. We also obtain a result linking the existence of circuits in the interaction graph to the existence of multiple attractors resp. an attractor with cardinality greater than one, which generalizes the corresponding statement in [10]. Both results demonstrate possibilities to study the network's dynamics without the explicit use of the state transition graph.

There are several starting points for future work. Although we have a good grasp on the behavior arising from circuits in the interaction graph, which are in some sense isolated, we have no clear understanding of the impact of intertwined circuits. In [3] the authors propose the concept of functionality context of a circuit, describing a set of states that ensure the effectiveness of the circuit interactions. Combining this idea with the notion of singular steady states may yield an approach to analyzing the behavior of networks containing intertwined circuits.

Besides extending the results for regulatory networks described by Boolean functions, a further goal is to generalize the approach to multi-valued, discrete functions, since they allow a refined modeling of bioregulatory networks.

References

1. Bernot, G., Comet, J.-P., Richard, A., Guespin, J.: Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. J. Theor. Biol. 229, 339–347 (2004)
2. Chaouiya, C., Remy, É., Mossé, B., Thieffry, D.: Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. In: First Multidisciplinary International Symposium on Positive Systems: Theory and Applications, POSTA 2003. LNCIS, vol. 294, pp. 119–126. Springer, Heidelberg (2003)
3. Naldi, A., Thieffry, D., Chaouiya, C.: Decision diagrams for the representation and analysis of logical models of genetic networks. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 233–247. Springer, Heidelberg (2007)
4. Remy, É., Mossé, B., Chaouiya, C., Thieffry, D.: A description of dynamical graphs associated to elementary regulatory circuits. Bioinform. 19, 172–178 (2003)
5. Remy, É., Ruet, P.: On differentiation and homeostatic behaviours of Boolean dynamical systems. In: Priami, C. (ed.) Transactions on Computational Systems Biology VIII. LNCS (LNBI), vol. 4780, pp. 92–101. Springer, Heidelberg (2007)
6. Remy, É., Ruet, P., Thieffry, D.: Graphic requirements for multistability and attractive cycles in a boolean dynamical framework (prépublication, 2005)
7. Richard, A., Comet, J.-P.: Necessary conditions for multistationarity in discrete dynamical systems. Rapport de Recherche (2005)
8. Richard, A., Comet, J.-P., Bernot, G.: R. Thomas' modeling of biological regulatory networks: introduction of singular states in the qualitative dynamics. Fundamenta Informaticae 65, 373–392 (2005)
9. Robert, F.: Discrete Iterations: A Metric Study. Springer Series in Computational Mathematics, vol. 6. Springer, Heidelberg (1986)
10. Siebert, H., Bockmayr, A.: Relating attractors and singular steady states in the logical analysis of bioregulatory networks. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) AB 2007. LNCS, vol. 4545, pp. 36–50. Springer, Heidelberg (2007)
11. Snoussi, E.H., Thomas, R.: Logical identification of all steady states: the concept of feedback loop characteristic states. Bull. Math. Biol. 55, 973–991 (1993)
12. Soulé, C.: Graphical requirements for multistationarity. ComPlexUs 1, 123–133 (2003)
13. Thomas, R., d'Ari, R.: Biological Feedback. CRC Press, Boca Raton (1990)
14. Thomas, R., Kaufman, M.: Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits. Chaos 11, 180–195 (2001)

Investigating Generic Methods to Solve Hopf Bifurcation Problems in Algebraic Biology

Thomas Sturm[1] and Andreas Weber[2]

[1] Universität Passau, 94030 Passau, Germany
sturm@uni-passau.de
[2] Institut für Informatik II, Universität Bonn, Römerstr. 164, 53117 Bonn, Germany
weber@cs.uni-bonn.de

Abstract. Symbolic methods for investigating Hopf bifurcation problems of vector fields arising in the context of algebraic biology have recently obtained renewed attention. However, the symbolic investigations have not been fully algorithmic but required a sequence of symbolic computations intervened with ad hoc insights and decisions made by a human. In this paper we discuss the use of generic methods to reduce questions on the existence of Hopf bifurcations in parameterized polynomial vector fields to quantifier elimination problems over the reals combined with simplification techniques available in REDLOG. We can reconstruct most of the results given in the literature within a few seconds of computation time. As no tedious hand computations are involved we presume that the use of these generic methods will be a useful tool for investigating other examples.

1 Introduction

Symbolic methods to investigate Hopf bifurcation problems of vector fields arising in the context of algebraic biology have recently obtained renewed attention [1,2,3]. A major reason for this attention is the relationship between Hopf bifurcation fixed points and the occurrence of oscillations. Although this relationship is subtle—we refer to [1,3] for nice introductions containing further references—the topic of investigating "oscillations" in systems for biologically or chemically relevant values of parameters is of such great interest that considerable work has been done to investigate various biological and chemical systems with respect to Hopf bifurcation fixed points, see e.g. [4,5,6,7,8] to mention only some.

There exist software packages such as AUTO[1] or XPPAUT[2], which locate Hopf bifurcations by means of numerical calculations. They allow one to evidence the existence of Hopf bifurcations. However, if one wants to prove that no Hopf bifurcation fixed point exists at all, then numeric methods fail in principle. Furthermore, if there are ranges of parameters for which there are Hopf bifurcations but these ranges are rather small, then numeric methods might not

[1] indy.cs.concordai.ca/auto/
[2] www.math.pitt.edu/~bard/xpp/xpp.html

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 200–215, 2008.

detect them—a possibility that is not only of theoretical interest in the context of algebraic biology and chemical reaction systems.

Whereas theoretically the problem is known to be decidable [9,2,3] the symbolic investigations carried out for specific parameterized polynomial vector fields arising from larger examples, e.g. the ones investigated in [1,2], have not been fully algorithmic up to now but required a sequence of symbolic computation intervened with ad hoc insights and decisions made by a human, and sometimes of sophisticated coordinate transforms. The aim of this paper is to demonstrate how to proceed fully algorithmically for the examples discussed there.

For this we proceed as follows: We use the method described by El Kahoui and Weber [9,10] to generate from the symbolic description of the respective ordinary differential equation a first-order formula in the language of ordered rings, where our domain is the real numbers. This is always possible if the vector field consists of polynomials in the variables and parameters.

If one suspects that there is no Hopf bifurcation fixed point or one just wants to assert that there is one, then one can apply quantifier elimination to the existential closure, i.e. all parameters are existentially quantified, of our generated formula. More generally, applying quantifier elimination to the original formula yields in principle a quantifier-free semi-algebraic description of the parameters for which Hopf bifurcation fixed points exist. In practice this latter variant does not finish within reasonable time at present. There is, however, an alternative, which provides at least one sample solution in the positive case; viz. extended quantifier elimination [11].

For the quantifier elimination we use REDLOG [12], which provides an automatic combination of virtual substitution methods [13] with partial CAD [14]. After noticing quite soon that most of our generated formulae were intractable by the regular elimination method, we developed a modified quantifier elimination procedure called *positive quantifier elimination*. This is based on the observation that in the considered examples, the variables as well as almost all of the parameters can be restricted to positive values. It has turned out that this knowledge greatly supports the elimination process. A current version of REDLOG including positive quantifier elimination is available for free download on the REDLOG website[3]. All computational examples discussed throughout this paper are contained in the online database REMIS there [15].

We can reconstruct the results of [1] in some seconds of computation time. Also the main example given in [2] could be handled in principle—without using the sophisticated coordinate transform used there.

2 Quantifier Elimination over the Reals

2.1 A Survey of Regular Quantifier Elimination

In order to summarize the basic idea of real quantifier elimination, we introduce first-order logic on top of polynomial equations and inequalities.

[3] `www.redlog.eu`

We consider multivariate polynomials $f(u, x)$ with rational coefficients, where $u = (u_1 \ldots, u_m)$ and $x = (x_1, \ldots, x_n)$. We call u *parameters* and we call x *variables*. Equations will be expressions of the form $f = 0$, inequalities are of the form $f \leq 0$, $f < 0$, $f \geq 0$, $f > 0$, or $f \neq 0$. Equations and inequalities are called *atomic formulae*. *Quantifier-free formulae* are Boolean combinations of atomic formulae by the logical operators "$\wedge$," "$\vee$," and "$\neg$." *Existential formulae* are of the form $\exists x_1 \ldots \exists x_n \psi(u, x)$, where ψ is a quantifier-free formula. Similarly, *universal formulae* are of the form $\forall x_1 \ldots \forall x_n \psi(u, x)$. A general (prenex) *first-order formula* has several alternating blocks of existential and universal quantifiers in front of a quantifier-free formula.

The real *quantifier elimination problem* can be phrased as follows: Given a formula φ, find a quantifier-free formula φ' such that both φ and φ' are equivalent in the domain of the real numbers. A procedure computing such a φ' from φ is called a real *quantifier elimination procedure*.

Quantifier elimination for an existential formula $\varphi(u) \equiv \exists x_1 \ldots \exists x_n \psi(u, x)$ has a straightforward geometric interpretation: Let

$$M = \{\, (u, x) \in \mathbb{R}^{m+n} \mid \psi(u, x) \,\},$$

and let $M' = \{\, u \in \mathbb{R}^m \mid \varphi(u) \,\}$. Then M' is the projection of M along the coordinate axes of the existentially quantified variables x onto the parameter space. Quantifier elimination yields a quantifier-free description of this projection.

Sets defined by first-order formulae are called *definable sets*. Sets defined by quantifier-free formulae are called *semi-algebraic sets*. Obviously, every semi-algebraic set is definable. The existence of a quantifier elimination procedure implies that, vice versa, every definable set is already semi-algebraic.

Obviously, quantifier elimination for existential formulae provides a test that determines the solvability of a parametric system of equations in dependence on the parameters. The procedure gives, however, no information on possible *solutions* of the input system. This point of view gives rise to the following generalization of quantifier elimination: Given an existential formula $\varphi \equiv \exists x_1 \ldots x_n \psi(u, x)$, we wish to compute a set

$$\Phi' = \left\{\, \left(\varphi'_k(u), \alpha_k(u)\right) \mid k \in K \,\right\}, \quad K \text{ finite},$$

which has the following properties:

1. The φ'_k are quantifier-free formulae. The α_k provide terms $t_1(u), \ldots, t_n(u)$ corresponding to the eliminated variables $x_1, \ldots, x_n$.
2. Define φ' as $\bigvee_{k \in K} \varphi'_k$. Then φ' is equivalent to φ in the reals. In other words, φ' is obtained from φ by quantifier elimination.
3. Fix real values for the parameters u: if φ and hence φ' holds, then there is some φ'_k which holds. The corresponding *answer* α_k is a sample point, which when *virtually substituted* for x satisfies ψ.

The notion of virtual substitution refers to the fact that the terms $t_1(u), \ldots, t_n(u)$ possibly contain some non-standard symbols like quotients, root expressions, or

infinitesimals, which can be substituted into quantifier-free formulae in such a way that they do not formally occur in the substitution result. An algorithm mapping φ to Φ' as described above is called an *extended* quantifier elimination procedure, or a quantifier elimination *with answer*.

2.2 Positive Quantifier Elimination

In the context of algebraic biology it is often known that *all* (or at least most) variables and parameters are positive. Let us assume first that all variables and parameters are positive. Such a global information greatly supports the quantifier elimination process by virtual substitution. We call the resulting procedure *positive quantifier elimination*. We do not go into technical details on this here but indicate a few major points, where positivity can be exploited. To start with, recall that the practical applicability of substitution methods depends crucially on efficient and powerful simplification of intermediate and final results [16]. These simplification methods can be considerably improved by positivity assumptions on all variables:

- There is an established simplification strategy which checks for terms that are *(strict) trivial squaresums* [16]. These are by definition sums of monomials with positive coefficients and even degree in all variables (and a positive absolute summand). Obviously, trivial squaresums are positive semi-definite, and strict trivial squaresums are positive definite. Thus corresponding atomic formulae can be evaluated to true or false or at least be simplified to equations or negated equations. On the assumption that all variables are positive, we need not check for positive degrees. For instance, we generally know that

  $$3x^2y^4 + 7x^2 \geq 0.$$

  With positive quantifier elimination, we may go further and even equivalently replace by "true" the atomic formula

  $$3xy + 7x > 0.$$

  Notice that technically we need only literally check for a minus sign in some canonical recursive or distributive representation of left hand side polynomials.
- The search for (generalized) trivial squaresums can be extended to testing irreducible factors of the occurring polynomials. For instance, in the positive case we can go

  $$ax^2 + axy^2 - bx^2 - by^2 \leq 0 \longleftrightarrow (ax - b)(x + y^2) \leq 0 \longleftrightarrow ax - b \leq 0.$$

  Since our positivity assumptions greatly increase our chance to discover such squaresums, it pays off in many cases to factorize more systematically. In order to optionally do so we have introduced into REDLOG a switch `rlsifaco` ("simplifier factorization with ordering relations"), which is going to play a role in the discussion of our computation examples later on.

Next, we illustrate two situations, where we can exploit positivity within the virtual substitution process itself:

- When virtually substituting into atomic formulae quotients with parametric denominators one must in general multiply with the square of the denominator in order to preserve signs. As an example consider

$$\big(ax - b > 0\big)\big[x /\!\!/ \tfrac{c}{y+z^2}\big] \equiv acy + acz^2 - by^2 - 2byz^2 - bz^4 > 0.$$

  With positive quantifier elimination one heuristically discovers non-negative denominators or at least non-negative factors based on strategies as sketched for simplification above. In our example, this allows to simply drop the then positive denominator:

$$\big(ax - b > 0\big)\big[x /\!\!/ \tfrac{c}{y+z^2}\big] \equiv ac - by - bz^2 > 0.$$

  At this point one possibly avoids subsequent degree violations for quantified variables in the denominator; in our example, z might be a quantified variable.
- There is a degree decreasing shift operation which essentially divides all exponents of quantified variables by the GCD of these exponents [17]. Doing so, one obviously has to take care of positivity, e.g., when switching between even and odd degrees. This operation can be simplified and slightly generalized on our positivity assumption. As a simple example consider the following transformation, where with positive quantifier elimination we need not add a condition $z \geq 0$:

$$\exists z(az^2 - bz^4 > 0 \wedge z^4 - 3z^2 < 0) \longleftrightarrow \exists z(az - bz^2 > 0 \wedge z^2 - 3z < 0).$$

For our set of input formulas discussed throughout this paper, it turns out that all optimizations discussed above become relevant at some point. For input problems from other fields of applications, positive quantifier elimination has not been systematically evaluated yet.

Notice that positive quantifier elimination as a concept is not at all restricted to virtual substitution methods. For instance, during the projection phase of partial CAD one could drop polynomials that are definite on positivity assumptions.

It is theoretically quite straightforward to equip positive quantifier elimination with an optional argument to pass a list of parameters and variables that are *not* known to be positive. This is, however, not yet implemented at present. We are now going to discuss how to alternatively deal with a parameter that is not known to be positive, say λ_1.

In applications, it might happen that the parameter occurs in a linear equation. In this special case, one can eliminate it by solving for λ_1 in one of the linear equations involving this parameter. Algorithmically, this strategy can be realized in a preprocessing step, e.g. the one realized by Brown and Groß [18], but is of course restricted to special cases.

One generally applicable procedure is to perform a case distinction on $\lambda_1 > 0$, $\lambda_1 < 0$, or $\lambda_1 = 0$. This results in three positive quantifier elimination runs, where in the two latter cases we substitute $\lambda_1 \leftarrow -\lambda_1$ and $\lambda_1 \leftarrow 0$, respectively. Iterating this procedure for other general parameters $\lambda_2, \ldots, \lambda_n$ yields 3^n many case distinctions, so that the case distinction is only feasible for small n, i.e. there are only few parameters not known to be positive.

There is another theoretically interesting option: For quantified non-positive variables we observe that every real number is a difference of two positive real numbers and go $\lambda_1 \leftarrow \lambda_1 - \lambda_1'$, where λ_1' is quantified in the same way as λ_1.

Note that both our substitution techniques sketched above are completely algorithmic. They can easily be implemented on top of any implementation of positive quantifier elimination.

3 Computation Examples

3.1 Models of Genetic Circuits

The examples investigated in [1] consist of a family of ordinary differential equations of the following form:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}G(t) &= \vartheta(\gamma_0 - G(t) - G(t)P(t)^n),\\
\frac{\mathrm{d}}{\mathrm{d}t}P(t) &= n\alpha(\gamma_0 - G(t) - G(t)P(t)^n) + \delta(M(t) - P(t)),\\
\frac{\mathrm{d}}{\mathrm{d}t}M(t) &= \lambda_1 G(t) + \gamma_0\mu - M(t),
\end{aligned} \tag{1}$$

where n is a natural number. We have renamed λ to λ_1 because `lambda` is obviously a bad choice for a variable name in Lisp systems. All variables and parameters except λ_1 are known to be positive. For a description of the model, from which this family of ordinary differential equations arise, we refer to [1].

Using a Maple library[4] containing the methods described in [9] we can generate the first order formulae stating the question on the existence of Hopf bifurcation fixed points (taking into account the known positivity conditions) by the Maple script in Figure 1.

Here we generate formulae up to $n = 10$ in a slight extension of the cases considered in [1]. There it is proved that no Hopf bifurcations exist for $n \leq 8$ and a Hopf bifurcations fixed point exists for $n = 9$.

In Figure 2 we present the generated first-order input formulae for $n = 2$ and $n = 9$. The system variables are renamed to `vv1`, `vv2`, `vv3` by the Maple library; as they occur in quantified form only, this renaming is of little concern. For better readability they are typeset as $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$. Apart from this, we literally give the formulae as they are output by Maple. They happen to contain some redundant parentheses.

[4] The current version of the library can be obtained at cg.cs.uni-bonn.de/project-pages/symbolicanalysis/

```
eq1n := diff(G(t),t)=(theta*(gamma0-G(t)-G(t)*P(t)^n));
eq2n := diff(P(t),t)=n*alpha*(gamma0-G(t)-G(t)*P(t)^n)+delta*(M(t)-P(t));
eq3n := diff(M(t),t)=lambda1*G(t)+gamma0*mu-M(t);
fcns1 := {G(t),P(t),M(t)};
params1 := [theta,alpha,gamma0,mu,lambda1];
paramcondlist := [theta>0, gamma0>0, mu>0, delta>0, alpha>0];
funccondlist := [G(t)>0, P(t)>0, M(t)>0];
lown:= 0; upn :=10;
for nn from lown to upn do
   eq11 := eval(subs(n=nn,eq1n));
   eq21 := eval(subs(n=nn,eq2n));
   eq31 := eval(subs(n=nn,eq3n));
   DEHopfexistence({eq11,eq21,eq31},
      fcns1,params1,funccondlist,paramcondlist);
od;
```

Fig. 1. A Maple script for generating the first-order input formulae $\varphi_2, \ldots, \varphi_{10}$ for Section 3.1

$$
\begin{aligned}
\varphi_2 \equiv\ & \exists \mathbf{v}_1 \exists \mathbf{v}_2 \exists \mathbf{v}_3 (((0 < \mathbf{v}_1 \wedge 0 < \mathbf{v}_3) \wedge 0 < \mathbf{v}_2) \wedge \\
& (((((((\vartheta(\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1\mathbf{v}_3^2) = 0 \wedge \\
& \lambda_1 \mathbf{v}_1 + \gamma_0 \mu - \mathbf{v}_2 = 0) \wedge \\
& 2\alpha(\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1\mathbf{v}_3^2) + \delta(\mathbf{v}_2 - \mathbf{v}_3) = 0) \wedge \\
& (0 < \vartheta\delta + \vartheta\mathbf{v}_3^2\delta + 2\lambda_1\vartheta\mathbf{v}_1\mathbf{v}_3\delta \wedge \\
& \vartheta\mathbf{v}_3^2 + 2\vartheta\delta + 8\vartheta\alpha\mathbf{v}_1\mathbf{v}_3 + 4\alpha\mathbf{v}_1\mathbf{v}_3\vartheta\delta + 4\alpha\mathbf{v}_1\mathbf{v}_3^3\vartheta\delta + 8\alpha\mathbf{v}_1\mathbf{v}_3\delta + \delta^2 + \\
& \quad \vartheta\delta^2 + 16\alpha^2\mathbf{v}_1^2\mathbf{v}_3^2 + \vartheta\mathbf{v}_3^2\delta^2 + 2\vartheta^2\mathbf{v}_3^2\delta + \vartheta^2\mathbf{v}_3^4\delta + \delta + \vartheta^2\delta + \vartheta^2 + \\
& \quad 2\vartheta^2\mathbf{v}_3^2 + \vartheta^2\mathbf{v}_3^4 + 4\alpha\mathbf{v}_1\mathbf{v}_3 + \vartheta + 2\vartheta\mathbf{v}_3^2\delta + 8\vartheta\mathbf{v}_3^3\alpha\mathbf{v}_1 - 2\lambda_1\vartheta\mathbf{v}_1\mathbf{v}_3\delta = 0)) \wedge \\
& 0 < \vartheta) \wedge 0 < \gamma_0) \wedge 0 < \mu) \wedge 0 < \delta \wedge 0 < \alpha)),
\end{aligned}
$$

$$
\begin{aligned}
\varphi_9 \equiv\ & \exists \mathbf{v}_2 \exists \mathbf{v}_1 \exists \mathbf{v}_3 (((0 < \mathbf{v}_1 \wedge 0 < \mathbf{v}_3) \wedge 0 < \mathbf{v}_2) \wedge \\
& (((((((\vartheta(\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1\mathbf{v}_3^9) = 0 \wedge \\
& \lambda_1 \mathbf{v}_1 + \gamma_0 \mu - \mathbf{v}_2 = 0) \wedge \\
& 9\alpha(\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1\mathbf{v}_3^9) + \delta(\mathbf{v}_2 - \mathbf{v}_3) = 0) \wedge \\
& (0 < \vartheta\delta + \vartheta\mathbf{v}_3^9\delta + 9\lambda_1\vartheta\mathbf{v}_1\mathbf{v}_3^8\delta \wedge \\
& 162\vartheta\mathbf{v}_3^{17}\alpha\mathbf{v}_1 + 162\vartheta\alpha\mathbf{v}_1\mathbf{v}_3^8 + 162\alpha\mathbf{v}_1\mathbf{v}_3^8\delta + \vartheta + 2\vartheta\mathbf{v}_3^9\delta + \vartheta^2\mathbf{v}_3^{18}\delta + \\
& \quad \vartheta\mathbf{v}_3^9 + 2\vartheta\delta + 81\alpha\mathbf{v}_1\mathbf{v}_3^8\vartheta\delta + 81\alpha\mathbf{v}_1\mathbf{v}_3^{17}\vartheta\delta + \delta^2 + \vartheta\delta^2 + \vartheta^2\delta + \vartheta^2 + \\
& \quad 2\vartheta^2\mathbf{v}_3^9 + \vartheta^2\mathbf{v}_3^{18} + 6561\alpha^2\mathbf{v}_1^2\mathbf{v}_3^{16} + 2\vartheta^2\mathbf{v}_3^9\delta + \delta + 81\alpha\mathbf{v}_1\mathbf{v}_3^8 + \vartheta\mathbf{v}_3^9\delta^2 - \\
& \quad 9\lambda_1\vartheta\mathbf{v}_1\mathbf{v}_3^8\delta = 0)) \wedge \\
& 0 < \vartheta) \wedge 0 < \gamma_0) \wedge 0 < \mu) \wedge 0 < \delta \wedge 0 < \alpha)).
\end{aligned}
$$

Fig. 2. The input formulae φ_2 and φ_9 generated by the Maple script in Figure 1. The Maple variables `vv1`, ..., `vv3` generated by the function `DEHopfexistence` in the script are typeset as $\mathbf{v}_1, \ldots, \mathbf{v}_3$ here.

Table 1. Applying regular quantifier elimination to the existential closures $\exists\varphi_2$, ..., $\exists\varphi_{10}$ of the formulae in Section 3.1. We systematically try several switch settings in REDLOG.

n	`rlsifaco`	`rlqeprecise`	`rlqevarseltry`	result	time (s)
2	○	○	○	SIGXCPU	> 600
3	○	○	○	SIGXCPU	> 600
4	○	○	○	SIGXCPU	> 600
4	○	○	●	false	113.26
4	●	○	●	false	84.10
5	○	○	○	SIGXCPU	> 600
6	○	○	○	SIGXCPU	> 600
6	●	○	●	false	170.10
7	○	○	○	SIGXCPU	> 600
8	○	○	○	SIGXCPU	> 600
8	○	○	●	false	407.63
8	●	○	●	false	346.77
9	○	○	○	SIGXCPU	> 600
10	○	○	○	SIGXCPU	> 600
10	●	○	●	true	543.82

As indicated in the introduction, we consider for now exclusively the existential closures $\exists\varphi_2$, ..., $\exists\varphi_{10}$ of our formulae. Then quantifier elimination yields either "true" or "false" depending on whether or not Hopf bifurcations exist. We are going to algorithmically treat the positive cases in more detail later on.

Applying Regular Quantifier Elimination. Using the regular quantifier elimination of REDLOG or QEPCAD with their default settings we could not decide any of the existential sentences. Systematically trying some REDLOG switches, we managed to determine special settings that deliver results also for $n = 4, 6, 8, 10$. Table 1 summarizes this experiment, where all rows with unsuccessful non-standard switch settings have been deleted. The keyword SIGXCPU in the result column indicates that the computation has been automatically interrupted after exceeding our chosen limit of 10 minutes of CPU time. The switch `rlsifaco` ("simplifier factorization with ordering relations") toggles the factorization of left hand side polynomials of ordering inequalities for the purpose of simplification; `rlqeprecise` ("quantifier elimination using precise test points") avoids to some extent the substitution of non-standard symbols with virtual substitution; `rlqevarseltry` ("quantifier elimintion variable selection try systematically") does not only apply the standard heuristics for choosing the next variable to be eliminated but additionally breaks tries by trying all best choices and continuing with the smallest result. By default all theses switches are off—for good reason, since this provides in general the by far best performance. Altogether, we consider these results not at all satisfactory: a regular user of some software must not be expected to run lengthy experiments with switch settings in order to hopefully obtain some results.

Table 2. Applying positive quantifier elimination to the existential closures $\underline{\exists}\varphi_2$, ..., $\underline{\exists}\varphi_{10}$ of the formulae in Section 3.1. We use the standard switch settings in REDLOG; `rlsifaco` is switched on, which is a general recommendation for positive quantifier elimination.

n	$\underline{\exists}\varphi_n$	$\underline{\exists}\varphi_n[\lambda_1 \leftarrow -\lambda_1]$	$\underline{\exists}\varphi_n[\lambda_1 \leftarrow 0]$	time (s)	$\underline{\exists}\varphi[\lambda_1 \leftarrow \lambda_1 - \lambda_1']$	time (s)
2	false	false	false	< 0.01	SIGXCPU	> 600
3	false	false	false	19.28	SIGXCPU	> 600
4	false	false	false	21.58	SIGXCPU	> 600
5	false	false	false	19.09	SIGXCPU	> 600
6	false	false	false	23.72	SIGXCPU	> 600
7	false	false	false	23.89	SIGXCPU	> 600
8	false	false	false	22.35	SIGXCPU	> 600
9	true	false	false	0.17	true	69.10
10	true	false	false	0.17	true	69.19

Applying Positive Quantifier Elimination. Our idea is now to improve the situation by making use of the observation that all variables and parameters except λ_1 are known to be positive. This observation has in fact been used also in [1] by solving for λ_1 in one of the linear equations involving this parameter, so that all remaining variables and parameters are known to be positive and then performing a "hand analysis."

Our approach avoids any hand analysis, because by using the generally applicable case distinction approach on λ_1, cf. Sect. 2.2, we can solve each of the examples in less than half a minute of computation time, cf. the left part of Table 2.

Our results for $n = 2, \ldots, 9$ confirm those in [1]; for our additional case $n = 10$, we discover the existence of a Hopf bifurcation fixed point. The computation times are the sums of times for all three respective eliminations. For this example, however, the cases $\lambda_1 < 0$ and $\lambda_1 = 0$ turn out to take no considerable time.

The right hand side part of Table 2 summarizes the alternative approach introducing instead of a case distinction a new variable λ_1' and substituting $\lambda_1 \leftarrow \lambda_1 - \lambda_1'$. It performs considerably worse on these examples. This might appear not surprising from the point of view that the complexity of our elimination procedure is exponential in the number of quantified variables. On the other hand, however, iterating case distinctions is exponential as well.

Obtaining Sample Points. For the cases in which a Hopf bifurcation fixed point exists the extended quantifier elimination [11] in REDLOG computes in addition a "sample point" fulfilling the existential quantifiers. For the following experiments, we turn on the switch `rlqeaprecise` ("quantifier elimination with answer using precise test points"), which is the analogue for extended quantifier elimination of `rlqeprecise` discussed above. It reduces the introduction of infinitesimals to some extent though not completely. In addition, we switch off `rlsiexpla` ("simplifier explode always") so that atomic formulae are not split by polynomial factorization unless the Boolean structure is preserved.

For the system with $n = 9$ we obtain within 0.11 s the following sample point:

$$\begin{aligned}
\alpha &= 1 \\
\gamma_0 &= \frac{-2 \cdot \sqrt{1180986} - 2187 \cdot \varepsilon_1 + 2187}{2187} \\
\delta &= 1 \\
\lambda_1 &= L(\varepsilon_1, \varepsilon_2) \\
\mu &= M(\varepsilon_1, \varepsilon_2) \\
\vartheta &= T(\varepsilon_1, \varepsilon_2) \\
\mathbf{v}_1 &= \frac{-2 \cdot \sqrt{1180986} - 2187 \cdot \varepsilon_1 + 2187}{43048908} \\
\mathbf{v}_2 &= 3 \\
\mathbf{v}_3 &= 3.
\end{aligned}$$

Here the $\varepsilon_1, \varepsilon_2$ are positive infinitesimals, which have to be interpreted as follows: There are $0 < \varepsilon_1, \varepsilon_2 \in \mathbb{R}$, which yield valid sample points, and moreover all positive choices less than ε_1 and ε_2, respectively, yield valid sample points too. The symbols L, M, and T denote rather complicated algebraic expressions, which we present in Appendix A.

In order to get an idea about approximate solutions we set $\varepsilon_1 = \varepsilon_2 = 0$, which yields:

$$\begin{aligned}
\gamma_0 &= 0.00618948546946, & \lambda_1 &= 9540696.11947, \\
\vartheta &= 0.0000571304095036, & \mathbf{v}_1 &= 0.000000314442464411.
\end{aligned}$$

For the case $n = 10$ the same procedure yields within 0.09 s the following approximate sample point, where again γ_0, λ_1, ϑ, and $\mathbf{v}_1$ are approximated by fixing two infinitesimals to 0:

$$\begin{aligned}
\alpha &= 1 \\
\gamma_0 &= 0.0100554964908 \\
\delta &= 1 \\
\lambda_1 &= 17617230.5528 \\
\mu &= 0 \\
\vartheta &= 0.0000211443608455 \\
\mathbf{v}_1 &= 0.000000170287832189 \\
\mathbf{v}_2 &= 3 \\
\mathbf{v}3 &= 3.
\end{aligned}$$

One can now systematically enumerate further solutions by adding to the input formulae conditions prohibiting for one or several variables or parameters the solutions already found.

```
eq11:=diff(x1(t),t)=k21*x1(t)^2*x2(t)+k46-k64*x1(t)-k34*x1(t)+k43*x3(t);
eq12:=diff(x2(t),t)=-k21*x1(t)^2*x2(t)+k56-k65*x2(t);
eq13:=diff(x3(t),t)=k34*x1(t)-k43*x3(t);
fcns1 := {x1(t),x2(t),x3(t)};
params1 := [k21,k46,k64,k34,k43,k56,k65];
paramcondlist1 := [k21>0, k46>0, k64>0, k34>0, k43>0, k56>0, k65>0];
funccondlist1 := [x1(t)>0, x2(t)>0, x3(t)>0];
DEHopfexistence({eq11,eq12,eq13},
   fcns1,params1,funccondlist1,paramcondlist1);
```

Fig. 3. A Maple script for generating the first-order input formula φ for Section 3.2

$$
\begin{aligned}
\varphi \equiv\ & \exists \mathbf{v}_3 \exists \mathbf{v}_2 \exists \mathbf{v}_1 (((0 < \mathbf{v}_1 \wedge 0 < \mathbf{v}_2) \wedge 0 < \mathbf{v}_3) \wedge \\
& ((((((((((k_{21}\mathbf{v}_1^2\mathbf{v}_2 + k_{46} - k_{64}\mathbf{v}_1 - k_{34}\mathbf{v}_1 + k_{43}\mathbf{v}_3 = 0 \wedge \\
& -k_{21}\mathbf{v}_1^2\mathbf{v}_2 + k_{56} - k_{65}\mathbf{v}_2 = 0) \wedge \\
& k_{34}\mathbf{v}_1 - k_{43}\mathbf{v}_3 = 0) \wedge \\
& (0 < k_{64}k_{65}k_{43} + k_{64}k_{21}\mathbf{v}_1^2k_{43} - 2k_{21}\mathbf{v}_1\mathbf{v}_2k_{65}k_{43} \wedge \\
& k_{21}\mathbf{v}_1^2k_{43}^2 + k_{21}^2\mathbf{v}_1^4k_{43} + k_{21}^2\mathbf{v}_1^4k_{34} + 2k_{34}k_{65}k_{43} + 2k_{64}k_{34}k_{65} + 2k_{64}k_{65}k_{43} + \\
& \quad k_{64}^2k_{21}\mathbf{v}_1^2 + k_{21}^2\mathbf{v}_1^4k_{64} + k_{34}^2k_{21}\mathbf{v}_1^2 + k_{34}k_{64}k_{43} + k_{64}k_{43}^2 + k_{65}k_{43}^2 + k_{64}k_{65}^2 + \\
& \quad k_{65}^2k_{43} + k_{34}k_{65}^2 + k_{64}^2k_{65} + k_{64}^2k_{43} + 2k_{64}k_{21}\mathbf{v}_1^2k_{43} + 2k_{34}k_{21}\mathbf{v}_1^2k_{43} + \\
& \quad 2k_{21}\mathbf{v}_1^2k_{64}k_{65} + 2k_{21}\mathbf{v}_1^2k_{65}k_{43} + 2k_{21}\mathbf{v}_1^2k_{34}k_{65} + 2k_{64}k_{34}k_{21}\mathbf{v}_1^2 - \\
& \quad 4k_{21}^2\mathbf{v}_1^3\mathbf{v}_2k_{43} - 2k_{21}^2\mathbf{v}_1^3\mathbf{v}_2k_{64} - 2k_{21}^2\mathbf{v}_1^3\mathbf{v}_2k_{34} + 4k_{21}^2\mathbf{v}_1^2\mathbf{v}_2^2k_{43} + 4k_{21}^2\mathbf{v}_1^2\mathbf{v}_2^2k_{65} - \\
& \quad 2k_{21}^2\mathbf{v}_1^3\mathbf{v}_2k_{65} - 2k_{21}\mathbf{v}_1\mathbf{v}_2k_{43}^2 - 2k_{21}\mathbf{v}_1\mathbf{v}_2k_{65}^2 + k_{34}^2k_{65} - 4k_{21}\mathbf{v}_1\mathbf{v}_2k_{65}k_{43} - \\
& \quad 4k_{21}\mathbf{v}_1\mathbf{v}_2k_{64}k_{65} - 4k_{21}\mathbf{v}_1\mathbf{v}_2k_{64}k_{43} - 4k_{21}\mathbf{v}_1\mathbf{v}_2k_{34}k_{65} - 2k_{34}k_{21}\mathbf{v}_1\mathbf{v}_2k_{43} = 0)) \wedge \\
& 0 < k_{21}) \wedge 0 < k_{46}) \wedge 0 < k_{64}) \wedge 0 < k_{34}) \wedge 0 < k_{43}) \wedge 0 < k_{56}) \wedge 0 < k_{65}))
\end{aligned}
$$

Fig. 4. The formula φ generated by the Maple script in Figure 3. The Maple variables `vv1`, ..., `vv3` generated by the function `DEHopfexistence` in the script are typeset as $\mathbf{v}_1, \ldots, \mathbf{v}_3$ here.

3.2 Mass Action Systems

As an example of a mass action system we consider the system investigated in [2, Example 2.1]. It is described by the following system of differential equations:

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}x_1(t) &= k_{21}x_1(t)^2x_2(t) + k_{46} - k_{64}x_1(t) - k_{34}x_1(t) + k_{43} \\
\frac{\mathrm{d}}{\mathrm{d}t}x_2(t) &= -k_{21}x_1(t)^2x_2(t) + k_{56} - k_{65}x_2(t) \\
\frac{\mathrm{d}}{\mathrm{d}t}x_3(t) &= k_{34}x_1(t) - k_{43}x_3(t). \qquad (2)
\end{aligned}
$$

Again only positive values for the variables are of interest. The automatic generation of a first-order formula stating the question on the existence of Hopf bifurcation fixed points and taking into account the positivity conditions is straightforward. Figure 3 shows our Maple script for this.

Table 3. Applying regular quantifier elimination to the existential closures $\underline{\exists}\varphi$ of the formula in Section 3.2. We systematically try several switch settings in REDLOG.

`rlsifaco`	`rlqeprecise`	`rlqevarseltry`	result	time (s)
○	○	○	SIGXCPU	> 600
○	○	●	true	261.72
○	●	○	true	3.03
○	●	●	true	1.54
●	○	○	SIGXCPU	> 600
●	○	●	true	264.73
●	●	○	true	3.35
●	●	●	true	1.79

Table 4. Applying positive quantifier elimination to the existential closures $\underline{\exists}\varphi$ of the formula in Section 3.2. We systematically try several switch settings in REDLOG.

`rlsifaco`	`rlqeprecise`	`rlqevarseltry`	result	time (s)
○	○	○	true	20.79
○	○	●	true	32.20
○	●	○	SIGXCPU	> 600
○	●	●	SIGXCPU	> 600
●	○	○	SIGXCPU	> 600
●	○	●	true	32.54
●	●	○	SIGXCPU	> 600
●	●	●	true	8.03

The script produces the first-order input formula in Figure 4, which gives the exact, i.e. both necessary and sufficient condition for the existence of a Hopf bifurcation fixed point.

The original literature [2], in contrast, does less: it first derives a necessary condition, and then in a another analysis a sufficient one. To illustrate the difference, notice that "true" is always necessary and "false" is always sufficient but neither of them is generally necessary and sufficient.

Our formulation uses the original coordinates and does not use a coordinate transform as in [2]. Of course, such a coordinate transform could also be realized with our approach in a preprocessing step.

Tables 3 and 4 show the timings and results for regular and for positive quantifier elimination, respectively, on the existential closure of φ. Most surprisingly, on this example regular quantifier elimination performs considerably better than its positive variant. We consider this a rare exception. Anyway it will not be completely ignored: We expect from a careful analysis of the various quantifier elimination runs some insights and ideas for improvements of the procedures. There is another annoying fact, which cannot be ignored: In the previous section we had found that switching on `rlsifaco` ("simplifier factorization with ordering relations") and switching off `rlqeprecise` ("quantifier elimination using precise test points") and `rlqevarseltry` ("quantifier elimination variable selection try

systematically") should be the default choice for positive quantifier elimination. The 5th row in Table 4 shows that this very choice fails on this particular example. Notice, however, that things look much better when experimentally switching on `rlqevarseltry`. This indicates that the unusual behavior is mainly caused by the fact that good variables orderings cannot be easily recognized.

Anyway, good news is that we can decide the existence of a Hopf bifurcation fixed point within a few seconds. As the answer is affirmative, also sample answer points can be computed. For this we use regular extended quantifier elimination with the most efficient switch settings given in the 4th row of Table 3. We obtain after 3.05 s:

$$k_{21} = 32, \quad k_{34} = 2/3, \quad k_{43} = 1, \quad k_{46} = 1/12, \quad k_{56} = 1,$$
$$k_{64} = 14/3, \quad k_{65} = 1/2, \quad \mathbf{v}_1 = 1/8, \quad \mathbf{v}_2 = 1, \quad \mathbf{v}_3 = 1/12.$$

So for this example, the switch `rlqeaprecise` is powerful enough to entirely avoid the introduction of infinitesimals. Again, further sample points can be enumerated by explicitly excluding in the input formula the ones already found.

4 Conclusions and Future Work

Using generic methods to reduce the Hopf bifurcation problem to a quantifier elimination problem and then using quantifier elimination methods available in REDLOG we can reconstruct most of the results given in the literature within less than a minute of computation time.

We have restricted ourselves to examples already treated in the literature by other symbolic methods. However, we presume that our generic methods are applicable not only to many other problems in theory but also in practice. As no tedious hand computations are involved the use of these generic methods will hopefully be a useful tool for investigating a wide variety of other examples. In future work we will test our generic methods for parametrically investigating Hopf bifurcation fixed points on examples treated in the literature by numerical computations or pure hand calculations, e.g. on the systems described in [4,5,6,19]. As there is the option to have some parameters fixed and only to investigate few others symbolically—still extending the possibilities of pure numeric investigations—one can also apply these techniques to larger systems, which cannot be expected to be amenable for a full parametric analysis, e.g. the systems described in [7,8].

On the quantifier elimination side, the great challenge is to improve the procedure such that the original elimination problems, in contrast to the existential closures, become tractable. There is actually good hope for success: In principle the elimination for the closure is a harder problem than that for the original formula, but we apparently benefit from the greater freedom in choosing the variable order for elimination. The fact that this is successful indicates that the problems are not inherently hard.

Acknowledgement

We are grateful to A. Dolzmann, C. Groß, A. Reitelmann, and U. Waldmann for helpful discussions.

References

1. Boulier, F., Lefranc, M., Lemaire, F., Morant, P., Ürgüplü, A.: On proving the absence of oscillations in models of genetic circuits. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) AB 2007. LNCS, vol. 4545, pp. 66–80. Springer, Heidelberg (2007)
2. Gatermann, K., Eiswirth, M., Sensse, A.: Toric ideals and graph theory to analyze Hopf bifurcations in mass action systems. Journal of Symbolic Computation 40(6), 1361–1382 (2005)
3. Niu, W., Wang, D.: Algebraic approaches to stability analysis of biological systems. Mathematics in Computer Science 1(3), 507–539 (2008)
4. Sensse, A., Hauser, M.J.B., Eiswirth, M.: Feedback loops for Shilnikov chaos: The peroxidase-oxidase reaction. The Journal of Chemical Physics 125, 014901, 1–12 (2006)
5. Fussmann, G.F., Ellner, S.P., Shertzer, K.W., Hairston, N.G.J.: Crossing the Hopf bifurcation in a live predator-prey system. Science 290(5495), 1358–1360 (2000)
6. Mincheva, M., Roussel, M.R.: Graph-theoretic methods for the analysis of chemical and biochemical networks. I. multistability and oscillations in ordinary differential equation models. Journal of Mathematical Biology 55(1), 61–86 (2007)
7. Novak, B., Pataki, Z., Ciliberto, A., Tyson, J.J.: Mathematical model of the cell division cycle of fission yeast. Chaos: An Interdisciplinary Journal of Nonlinear Science 11(1), 277–286 (2001)
8. Tyson, J.J., Chen, K., Novak, B.: Network Dynamics and Cell Physiology. Nat. Rev. Mol. Cell Biol. 2(12), 908–916 (2001)
9. El Kahoui, M., Weber, A.: Deciding Hopf bifurcations by quantifier elimination in a software-component architecture. Journal of Symbolic Computation 30(2), 161–179 (2000)
10. Weber, A.: Quantifier elimination on real closed fields and differential equations. In: Löwe, B. (ed.) Algebra, Logic, Set Theory – Festschrift für Ulrich Felgner zum 65. Geburtstag. Studies in Logic, vol. 4, pp. 291–315. College Publications (2007)
11. Weispfenning, V.: Simulation and optimization by quantifier elimination. Journal of Symbolic Computation 24(2), 189–208 (1997); Special issue on applications of quantifier elimination
12. Dolzmann, A., Sturm, T.: Redlog: Computer algebra meets computer logic. ACM SIGSAM Bulletin 31(2), 2–9 (1997)
13. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. Applicable Algebra in Engineering Communication and Computing 8(2), 85–101 (1997)
14. Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. Journal of Symbolic Computation 12(3), 299–328 (1991)
15. Sturm, T.: Redlog online resources for applied quantifier elimination. Acta Academiae Aboensis, Ser. B 67(2), 177–191 (2007)

16. Dolzmann, A., Sturm, T.: Simplification of quantifier-free formulae over ordered fields. Journal of Symbolic Computation 24(2), 209–231 (1997)
17. Dolzmann, A., Sturm, T., Weispfenning, V.: A new approach for automatic theorem proving in real geometry. Journal of Automated Reasoning 21(3), 357–380 (1998)
18. Brown, C.W., Groß, C.: Efficient preprocessing methods for quantifier elimination. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2006. LNCS, vol. 4194, pp. 89–100. Springer, Heidelberg (2006)
19. Sun, M., Tian, L., Yin, J.: Hopf bifurcation analysis of the energy resource chaotic system. International Journal of Nonlinear Science 1(1), 49–53 (2006)

A Exact Solutions with Infinitesimals for Section 3.1

$$
\begin{aligned}
L(\varepsilon_1,\varepsilon_2) = \Big(&-200156684335646976\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\cdot\varepsilon_2\\
&-22877954716428\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\\
&+5083989936984\cdot\sqrt{1180986}+5559342996092004\cdot\varepsilon_1\\
&+23640506047897342779392\cdot\varepsilon_2^2+5404230477062468352\cdot\varepsilon_2\\
&-5216187623191776\Big)/\\
&\Big(1062882\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{393662}\\
&+387420489\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\cdot\varepsilon_1\\
&-387420489\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\\
&-172186884\cdot\sqrt{1180986}\cdot\varepsilon_1-83691328896\cdot\sqrt{1180986}\cdot\varepsilon_2\\
&+166872690\cdot\sqrt{1180986}-94143178827\cdot\varepsilon_1^2\\
&-91516468147776\cdot\varepsilon_1\cdot\varepsilon_2+182475286515\cdot\varepsilon_1\\
&+91516468147776\cdot\varepsilon_2-181313497440\Big)
\end{aligned}
$$

$$
\begin{aligned}
M(\varepsilon_1,\varepsilon_2) = \Big(&61010978765184\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{393662}\cdot\varepsilon_2\\
&+2223850175990956 8\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\cdot\varepsilon_1\cdot\varepsilon_2\\
&-2223850175990956 8\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\cdot\varepsilon_2\\
&-2402002240184651776\cdot\sqrt{1180986}\cdot\varepsilon_2^2\\
&-2626589449641916717056\cdot\varepsilon_1\cdot\varepsilon_2^2\\
&+2626589449641916717056\cdot\varepsilon_2^2\Big)/\\
&\Big(4649045868\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{393662}\cdot\varepsilon_1
\end{aligned}
$$

$$
\begin{aligned}
&-4649045868\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{393662}\\
&+847288609443\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\cdot\varepsilon_1^2\\
&-1694577218886\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\cdot\varepsilon_1\\
&+1684121117211\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\\
&-564859072962\cdot\sqrt{1180986}\cdot\varepsilon_1^2\\
&-366065872591104\cdot\sqrt{1180986}\cdot\varepsilon_1\cdot\varepsilon_2\\
&+1106473861368\cdot\sqrt{1180986}\cdot\varepsilon_1+366065872591104\cdot\sqrt{1180986}\cdot\varepsilon_2\\
&-727577567910\cdot\sqrt{1180986}-205891132094649\cdot\varepsilon_1^3\\
&-200146515839186112\cdot\varepsilon_1^2\cdot\varepsilon_2+604964583702954\cdot\varepsilon_1^2\\
&+400293031678372224\cdot\varepsilon_1\cdot\varepsilon_2-1202306669284833\cdot\varepsilon_1\\
&-397823091334329024\cdot\varepsilon_2+790681240245960\Big)
\end{aligned}
$$

$$
\begin{aligned}
T(\varepsilon_1,\varepsilon_2)=\Big(&-6561\cdot\sqrt{\varepsilon_1}\cdot\sqrt{4\cdot\sqrt{1180986}+2187\cdot\varepsilon_1}\cdot\sqrt{3}\\
&+1458\cdot\sqrt{1180986}+1594323\cdot\varepsilon_1\\
&+1549839424\cdot\varepsilon_2-1495912\Big)/1549839424.
\end{aligned}
$$

Constructing a Knowledge Base for Gene Regulatory Dynamics by Formal Concept Analysis Methods

Johannes Wollbold[1,2], Reinhard Guthke[2], and Bernhard Ganter[1]

[1] University of Technology, Institute of Algebra, Dresden, Germany
jwollbold@gmx.de
http://www.math.tu-dresden.de/alg/algebra.html
[2] Leibniz Institute for Natural Product Research and Infection Biology, Hans-Knöll-Institute (HKI) Jena, Germany

Abstract. Our aim is to build a set of rules, such that reasoning over temporal dependencies within gene regulatory networks is possible. The underlying transitions may be obtained by discretizing observed time series, or they are generated based on existing knowledge, e.g. by Boolean networks or their nondeterministic generalization. We use the mathematical discipline of formal concept analysis (FCA), which has been applied successfully in domains as knowledge representation, data mining or software engineering. By the *attribute exploration* algorithm, an expert or a supporting computer program is enabled to decide about the validity of a minimal set of implications and thus to construct a sound and complete knowledge base. From this all valid implications are derivable that relate to the selected properties of a set of genes. We present results of our method for the initiation of sporulation in *Bacillus subtilis*. However the formal structures are exhibited in a most general manner. Therefore the approach may be adapted to signal transduction or metabolic networks, as well as to discrete temporal transitions in many biological and nonbiological areas.

Keywords: Complete lattices, reasoning, temporal logic, gene expression.

1 Introduction

As the mathematical methodology of formal concept analysis (FCA) is little known within systems biology, we give a short overview of its history and purposes. During the early years 1980, FCA emerged within the community of set and order theorists, algebraists and discrete mathematicians. Its first aim was to find a new, concrete and meaningful approach to the understanding of complete lattices (ordered sets such that for every subset the supremum and the infimum exist). The following discovery showed to be very fruitful: Every complete lattice is representable as a hierarchy of concepts, which were conceived as sets of objects sharing a maximal set of attributes. This paved the way for using the developed field of lattice theory for a transparent and complete representation of

K. Horimoto et al. (Eds.): AB 2008, LNCS 5147, pp. 230–244, 2008.

very different types of knowledge. FCA was inspired by the pedagogue Hartmut von Hentig [7] and his program of restructuring sciences, with a view to interdisciplinary collaboration and democratic control. The philosophical background goes back to Charles S. Peirce (1839 - 1914), who condensed some of his main ideas to the pragmatic maxim:

> *Consider what effects, that might conceivably have practical bearings, we conceive the objects of our conception to have. Then, our conception of these effects is the whole of our conception of the object.* [14, 5.402]

In that tradition, FCA aims at unfolding the observable, elementary properties defining the objects subsumed by scientific concepts. If applied to temporal transitions, effects of homogeneous classes of states can be modeled and predicted in a clear and concise manner. Thus FCA seems to be appropriate to describe causality - and the limits of its understanding.

At present, FCA is a richly developed mathematical theory, and there are practical applications in various fields as data and text mining, knowledge management, semantic web, software engineering or economics [3]. FCA has been used for the analysis of gene expression data in [2] and [13], but this is the first approach of applying it to model (gene) regulatory networks. The mathematical framework of FCA is very general and open, such that multifarious refinements are possible, according to current approaches of modeling dynamics within systems biology. On the other hand, we developed a formal structure for general discrete temporal transitions. They occur in a variety of domains: control of engineering processes, development of the values of variables or objects in a computer program, change of interactions in social networks, a piece of music, etc.

In this paper, however, the examples are uniquely biological. The purpose is to construct a knowledge base for reasoning about temporal dependencies within gene regulatory or signal transduction networks, by the *attribute exploration* algorithm: For a given set of interesting properties, it builds a sound, complete and nonredundant knowledge base. This minimal set of rules has to be checked by an expert or a computer program, e.g. by comparison of knowledge based predictions with data.

Since there exist relatively fixed thresholds of activation for many genes, it is a common abstraction to consider only two expression levels *off* and *on*. The classical approach of Boolean networks [8] is able to capture essential dynamic aspects of regulatory networks. Our present work is based on it, which also makes it possible to use mathematical and logical derivations for deciding many rules automatically and for a scaling up to larger networks. Nevertheless, the introduction of more fine-grained expression levels remains possible, e.g. in the sense of qualitative reasoning [11]. Further, this work is influenced by computation tree logic [1], automata theory and a FCA modeling of temporal transitions in [15]. Temporal concept analysis as developed by K.E. Wolff [3, pp. 127–148] is more directed toward a structured visualization of experimental time series then toward temporal logic. We applied it to the analysis of gene expression data in [19].

In Section 2, the general mathematical framework will be developed. Section 3 gives results for a *B. subtilis* Boolean network. In Section 4, we will discuss the potential of the method and make some proposals for improving it by solving mathematical problems which have emerged from the applications.

2 Methods

2.1 Fundamental Structures of Formal Concept Analysis

One of the classical aims of FCA is the structured, compact but complete visualization of a data set by a conceptual hierarchy. We briefly introduce its basic definitions; for an easy example see `http://www.upriss.org.uk/fca/fca.html`

Definition 1. *A **formal context** (G, M, I) defines a relation $I \subseteq G \times M$ between objects from a set G and attributes from a set M. The set of the attributes common to all objects in $A \subseteq G$ is denoted by the $'$-operator:*

$$A' := \{m \in M |\, gIm \text{ for all } g \in A\}.$$

The set of the objects sharing all attributes in $B \subseteq M$ is

$$B' := \{g \in G |\, gIm \text{ for all } m \in B\}.$$

Definition 2. *A **formal concept** of the context (G, M, I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is the **extent**, B the **intent** of the concept (A, B).*

Thus a formal context (G, M, I) is a special, but universally applicable type of a data table, provided with two operators $\mathcal{C}_M : \mathcal{P}(M) \to \mathcal{P}(M)$, $B \subseteq M \mapsto B''$ and $\mathcal{C}_G : \mathcal{P}(G) \to \mathcal{P}(G)$, $A \subset G \mapsto A''$. It is easy to see that they are closure operators, with the properties monotony, extension and idempotency [4, Definition 14]. It follows that the set of all extents resp. intents of a formal context is a closure system, i.e. it is closed under intersections [4, Theorem 1].

Formal concepts can be ordered by set inclusion of the extents or - dually, with the inverse order relation - of the intents. With this order, the set of all concepts of a given formal context is a complete lattice [4, Theorem 3] (Figure 1).

During the interactive *attribute exploration* algorithm [4, p. 85ff.], an expert is asked about the general validity of basic implications $A \to B$ between the attributes of a given formal context (G, M, I). An implication has the meaning: "If an object $g \in G$ has all attributes $a \in A \subseteq M$, then it has also all attributes $b \in B \subseteq M$." If the expert denies, s/he must provide a counterexample, i.e. a new object of the context. If s/he accepts, the implication is added to the *stem base* of the - possibly enlarged - context. A theorem by Duquenne-Guiges [4, Theorem 8] ensures that every implication semantically valid in the underlying formal context can be derived syntactically from this minimal set by the Armstrong rules [4, Proposition 21]. In many applications, one is merely interested in the implicational logic of a given formal context, and there is no need for an expert to confirm the implications.

2.2 Constructing the Knowledge Base – Summary of the Method

We start with two sets:

- The universe E. The elements of E represent the **entities** of the world which we are interested in.
- The set F (**fluents**) denotes changing properties of the entities.

A **state** $\varphi \in G$ is an assignment of values in F to the variables $e \in E$, hence it is defined by a specific choice of attributes $m \in M \subseteq E \times F$.[1] By means of a state context (Definition 3, Table 3 left part), temporal data can be translated into the language of FCA. The dynamics is modeled by a binary relation $R \subseteq G \times G$ on the set of states, which gives rise to a transition context $\mathbb{K}$ (Definition 4, Table 1): the objects are transitions (elements of the relation) and the attributes the values of the entities defining the input and output state of a transition. This data table may reflect observations repeated at different time points, or the transitions may be generated by a dynamic model. As to the latter, we are focusing here on Boolean networks, i.e sets of Boolean functions for each entity (Definition 5).

It is promising to consider the transitive closure of R. Objects of the transitive context $\mathbb{K}_t$ (Definition 6, Table 2) then are pairs of states such that the output state emerges from the input state by some transition sequence of arbitrary length. Finally we extend the state context $\mathbb{K}_s$ by the temporal attributes always(m), eventually(m) and never(m), which are determined by the given transitions (Definition 3, Table 3).

The defined mathematical structures may be used in various ways. For instance, one could evaluate - i.e. generalize implications or reject them supposing outliers or by reason of special conditions - experimental time series by comparison with existing knowledge. Our general procedure is the following:

1. Discretize a set of time series of gene expression measurements and transform it to an observed transition context $\mathbb{K}^{obs}$.
2. For a set of interesting genes, translate interactions from biological literature and databases into a Boolean network.
3. Construct the transition context $\mathbb{K}$ by a simulation starting from a set of states, e.g. the initial states of $\mathbb{K}^{obs}$ or all states (for small networks).
4. Derive the respective transitive contexts $\mathbb{K}_t$ and $\mathbb{K}_t^{obs}$.
5. Perform attribute exploration of $\mathbb{K}_t$. Decide about an implication $A \to B$, $A, B \subseteq M$, by checking its validity in $\mathbb{K}_t^{obs}$ and/or by searching for supplementary knowledge. Possibly provide a counterexample from $\mathbb{K}_t^{obs}$.
6. Answer queries from the modified context $\mathbb{K}_t$ and from its stem base.

In step 5., automatic decision criteria could be thresholds of support $q = |(A \cup B)'|$ and confidence $p = \frac{|(A \cup B)'|}{|A'|}$ for an implication in $\mathbb{K}_t^{obs}$. A weak criterion

[1] Thus - as usual - states with the same variable values are identified. It would also be possible to distinguish them as situations by introducing a new attribute, e.g. "time interval".

is to reject only implications with support 0 (but if no object in $\mathbb{K}_t^{obs}$ has all attributes from A, the implication is not violated). In [18], a strong criterion has been applied: implications of $\mathbb{K}_t$ had to be valid also in the observed context ($p = 1$). This is equivalent to an exploration of the union of the two contexts.

In Section 3 we will analyse pure knowledge based simulations; the validation by data and experimental literature had been done before in [5]. For that reason, in step 5. the stem base is computed automatically, without further confirmation by an expert.

Step 2. could be supported by text mining software. Then attribute exploration provides strong criteria of validation. We implemented the steps 1., 3. and 4. in R [www.r-project.org]. For step 5., we used the Java tool Concept Explorer [http://sourceforge.net/projects/conexp]. The output was translated with R into a PROLOG knowledge base. The R scripts are available on request.

2.3 Definition of the Relevant Formal Contexts

With given sets E and F, we define a state as a map $\varphi : E \to F$. To explore static features of states, the following formal context[2] is defined.

Definition 3. *Given two sets E (entities) and F (fluents), a **state context** is a formal context (G, M, I) with $G \subseteq F^E := \{\varphi : E \to F\}$ and $M \subseteq E \times F$; its relation I is given as $\varphi\, I\, (e, f) \Leftrightarrow \varphi(e) = f$, for all $\varphi \in G, e \in E$ and $f \in F$.*

Definition 4. *Given a state context (G, M, I) and a relation $R \subseteq G \times G$, a **transition context** $\mathbb{K}$ is the context $(R, M \times \{in, out\}, \tilde{I})$ with the property*

$$\forall i \in \{in, out\}\colon\ (\varphi^{in}, \varphi^{out})\tilde{I}(e, f, i) \Leftrightarrow \varphi^i(e) = f. \tag{1}$$

Transitions may be generated by a Boolean network:

Definition 5. *Let E be an arbritray set of entities, $F := \{0, 1\}$ (fluents), and states $G \subseteq F^E$. Then a transition function $F^E \to F^E$ is called a **Boolean network**.*

We will identify the elements of F with $-, +$ or *off, on* respectively. This definition is subsumed by the definition of a dynamic network in [12, Definition p. 34], with a set of variables E and state sets $X_1 = ... = X_n = F$. We use a parallel update schedule, i.e. the order relation on E is empty. Boolean networks may be generalized in order to include nondeterminism; then different output states φ^{out} are generated from a single input state φ^{in} (see Section 3, compare [18]).

Definition 6. *A transition context $\mathbb{K}$ with a transitively closed relation $t(R) \subseteq G \times G$ is called a **transitive context** $\mathbb{K}_t$.*

[2] It is equivalent to a "many-valued context" with "nominal scale" [4, Section 1.3], [18, p. 123]. For better readability, we draw the contexts in the latter form (Table 3). Deriving a one-valued context according to Definition 3 is obvious: each many-valued attribute e is replaced by $\{(e, f) |\ f \in F\})$, e.g. SigA by SigA.off and SigA.on. If an attribute e takes exactly one of these values, negation of *on* and *off* is expressed. Other kinds of scaling like the *interordinal scale* could be interesting, if there are more than two levels ($|F| > 2$).

Table 1. A transition context for the states of Table 3, with all attributes that are changing during the small simulation, as well as Spo0A and Spo0AP

Transition	$\mathbf{KinA}^{in}$	$\mathbf{Spo0A}^{in}$	$\mathbf{Spo0AP}^{in}$	$\mathbf{AbrB}^{in}$	$\mathbf{Spo0E}^{in}$	$\mathbf{SigH}^{in}$	$\mathbf{Hpr}^{in}$	$\mathbf{KinA}^{out}$	$\mathbf{Spo0A}^{out}$	$\mathbf{Spo0AP}^{out}$	$\mathbf{AbrB}^{out}$	$\mathbf{Spo0E}^{out}$	$\mathbf{SigH}^{out}$	$\mathbf{Hpr}^{out}$
$(\varphi_0^{in}, \varphi_1^{out})$	-	+	-	-	-	-	+	-	+	-	+	+	+	-
$(\varphi_1^{in}, \varphi_2^{out})$	-	+	-	+	+	+	-	+	+	-	-	-	-	+
$(\varphi_2^{in}, \varphi_1^{out})$	+	+	-	-	-	-	+	-	+	-	+	+	+	-

Table 2. The transitive context derived from the transition context of Table 1

Transition	$\mathbf{KinA}^{in}$	$\mathbf{Spo0A}^{in}$	$\mathbf{Spo0AP}^{in}$	$\mathbf{AbrB}^{in}$	$\mathbf{Spo0E}^{in}$	$\mathbf{SigH}^{in}$	$\mathbf{Hpr}^{in}$	$\mathbf{KinA}^{out}$	$\mathbf{Spo0A}^{out}$	$\mathbf{Spo0AP}^{out}$	$\mathbf{AbrB}^{out}$	$\mathbf{Spo0E}^{out}$	$\mathbf{SigH}^{out}$	$\mathbf{Hpr}^{out}$
$(\varphi_0^{in}, \varphi_1^{out})$	-	+	-	-	-	-	+	-	+	-	+	+	+	-
$(\varphi_0^{in}, \varphi_2^{out})$	-	+	-	-	-	-	+	+	+	-	-	-	-	+
$(\varphi_1^{in}, \varphi_1^{out})$	-	+	-	+	+	+	-	-	+	-	+	+	+	-
$(\varphi_1^{in}, \varphi_2^{out})$	-	+	-	+	+	+	-	+	+	-	-	-	-	+
$(\varphi_2^{in}, \varphi_1^{out})$	+	+	-	-	-	-	+	-	+	-	+	+	+	-
$(\varphi_2^{in}, \varphi_2^{out})$	+	+	-	-	-	-	+	+	+	-	-	-	-	+

Definition 7. *A **state context** $\mathbb{K} = (G, M, I)$ is **extended** to a formal context $\mathbb{K}_s = (G, M \cup T, I \cup I_T)$ by a set of temporal attributes $T := \{always(m) | \, m \in M\} \cup \{never(m) | \, m \in M\} \cup \{eventually(m) | \, m \in M\}$. Let $\varphi^{in} \in G, m = (e, f)$, $e \in E$, $f \in F$, and $t(R) \subseteq G \times G$ a transitively closed relation. The relation I_T of $\mathbb{K}_s$ then is defined as follows:*

$$\varphi^{in} \; I_T \; always(m) \Leftrightarrow \forall (\varphi^{in}, \varphi^{out}) \in t(R) : \varphi^{out}(e) = f$$
$$\varphi^{in} \; I_T \; never(m) \Leftrightarrow \forall (\varphi^{in}, \varphi^{out}) \in t(R) : \varphi^{out}(e) \neq f$$
$$\varphi^{in} \; I_T \; eventually(m) \Leftrightarrow \exists (\varphi^{in}, \varphi^{out}) \in t(R) : \varphi^{out}(e) = f$$

For $B \subseteq T$, set $always(B) := \{\{always(b_1), ..., always(b_i)\} \mid b_1, ..., b_i \in B\}$, and analogously $never(B)$ and $eventually(B)$.

The attributes will be abbreviated to alw(m), nev(m) and ev(m). In a nondeterministic setting, alw(m) and nev(m) refer to all possible transition paths starting from φ^{in}, ev(m) to the existence of a path.

2.4 Dependency of Contexts and Background Knowledge

In the following we will present first mathematical results that can improve computability; they are not necessary for the understanding of the application

Table 3. *Left part*: A state context corresponding to a simulation starting from a *B. subtilis* state without nutritional stress (see Section 3.1, [16, Table 4]). +: on, -: off. *Right part*: extension by temporal attributes. Here they are the same for all states, since these reach the *attractor* (limit state cycle) $\{\varphi_1, \varphi_2\}$ after at most one time step.

State	$\mathbf{KinA}^{in}$	$\mathbf{Spo0A}^{in}$	$\mathbf{Spo0AP}^{in}$	$\mathbf{AbrB}^{in}$	$\mathbf{Spo0E}^{in}$	$\mathbf{SigH}^{in}$	$\mathbf{Hpr}^{in}$	**ev(KinA)**	**alw(KinA)**	**nev(Spo0AP)**	**ev(AbrB)**	**alw(AbrB)**	**ev(Hpr)**	...
φ_0^{in}	-	+	-	-	-	-	+	x		x	x		x	
φ_1^{in}	-	+	-	+	+	+	-	x		x	x		x	
φ_2^{in}	+	+	-	-	-	-	+	x		x	x		x	

in Section 3. By entering background knowledge (not necessarily implications) prior to an attribute exploration, the algorithm may be shortened considerably [3, pp. 101–113]. We searched for first order logic background formula in order to use the results of an attribute exploration for the exploration of the next context in the hierarchy. Then the implications of the latter context are derivable from this background knowledge and a reduced set of new implications. Also during the exploration of one context, implications can be decided automatically based on already accepted implications. In this way the expert is enabled to concentrate on really interesting hypotheses. Thus, the implications of a state context hold in the input and output part of the corresponding transition context (for an example see [15, p. 149f.]). Related to transitive and extended state contexts, the subsequent result holds:

Proposition 1. *Let* $\mathbb{K}_s = (G, M \cup T, I \cup I_T)$ *an extended state context. Suppose the relation* $t(R) \subseteq G \times G$ *is the object set of the transitive context* $\mathbb{K}_t = (t(R), M \times \{in, out\}, \tilde{I})$*. Then the following entailments between implications of both contexts are valid:*

$$B^{in} \to m^{out} \text{ in } \mathbb{K}_t \equiv B \to \textit{always}(m) \text{ in } \mathbb{K}_s \tag{2}$$

$$B^{in} \to m^{out} \text{ in } \mathbb{K}_t \models B \to \textit{eventually}(m) \text{ in } \mathbb{K}_s \tag{3}$$

$$B^{out} \to m^{out} \text{ in } \mathbb{K}_t \models \textit{always}(B) \to \textit{always}(m) \text{ in } \mathbb{K}_s \tag{4}$$

$$B^{out} \to m^{out} \text{ in } \mathbb{K}_t \models \textit{eventually}(B) \to \textit{eventually}(m) \text{ in } \mathbb{K}_s \tag{5}$$

$$B^{in} \cup m^{out} \to \bot \text{ in } \mathbb{K}_t \models B \to \textit{never}(m) \text{ in } \mathbb{K}_s \tag{6}$$

If the latter implication does not follow from the stem base of $\mathbb{K}_t$*, this is equivalent to* $B \to \textit{eventually}(m)$ *in* $\mathbb{K}_s$*.*

Proof. The proofs are straightforward from the definitions. □

In order to get a complete overview on valid entailments, as a first step we performed rule exploration [20] of the following **test context**, i.e. exploration

of Horn rules instead of implications, thus variables are admitted: The objects are all possible $\mathbb{K}_t$ respectively the corresponding $\mathbb{K}_s$, and the attributes are the following classes of implications with "homogeneous" premises. Then the explored rules for implications correspond to entailments valid for the semantics given by the objects, the transitive contexts. The sets are nonempty subsets of M, $m = (e, f)$, $f \in F := \{0, 1\}$, and $m \in B_0, C_0$ $(B_1, C_1) \Rightarrow \varphi(e) = 0$ (1). We suppose that all states and transitions are completely defined.

1. $B^{in} \to C^{in}$
2. $B^{in} \to C_0^{out} \equiv B^{in} \to \mathrm{nev}(C_1)$
3. $B^{in} \to C_1^{out} \equiv B^{in} \to \mathrm{alw}(C_1)$
4. $B^{in} \to \mathrm{ev}(C_1)$
5. $B_0^{out} \to C^{in}$
6. $B_1^{out} \to C^{in}$
7. $\mathrm{ev}(B_1) \to C^{in}$
8. $\mathrm{alw}(B_1) \to C^{in}$
9. $\mathrm{nev}(B_1) \to C^{in}$
10. $B_0^{out} \to C_0^{out}$
11. $B_0^{out} \to C_1^{out}$
12. $B_1^{out} \to C_0^{out}$
13. $B_1^{out} \to C_1^{out}$
14. $\mathrm{ev}(B_1) \to \mathrm{ev}(C_1)$
15. $\mathrm{ev}(B_1) \to \mathrm{alw}(C_1)$
16. $\mathrm{ev}(B_1) \to \mathrm{nev}(C_1)$
17. $\mathrm{alw}(B_1) \to \mathrm{ev}(C_1)$
18. $\mathrm{alw}(B_1) \to \mathrm{alw}(C_1)$
19. $\mathrm{alw}(B_1) \to \mathrm{nev}(C_1)$
20. $\mathrm{nev}(B_1) \to \mathrm{ev}(C_1)$
21. $\mathrm{nev}(B_1) \to \mathrm{alw}(C_1)$
22. $\mathrm{nev}(B_1) \to \mathrm{nev}(C_1)$

The equivalences in 2. and 3. follow from Proposition 1(2). Since the implications comprising input attributes are independent from those related only to output attributes, rule exploration was performed (almost) independently for the first 9 and the remaining 13 implications. Results for the second part are shown here.

The exploration started from a hypothetical context as single object of the test context, where no implications were valid. Before, we had added 25 known entailments as background rules (BR), like those of Proposition 1 or following from the definitions, like $\mathrm{alw}(B_1) \to \mathrm{ev}(B_1)$. A counterexample represents a significant class of contexts. They had to be chosen carefully, since an object not having its maximal attribute set might preclude a valid entailment.[3] The exploration resulted in the following stem base of only 14 entailments. Most of them are background rules (they are accepted automatically during the exploration), but not all of these are needed in order to derive all valid entailments between the chosen implications. This demonstrates the effectivity and minimality of the algorithm. Entailments 5., 6., 7. and 10. were newly found.

1. $\mathrm{nev}(B_1) \to \mathrm{alw}(C_1) \models \mathrm{nev}(B_1) \to \mathrm{ev}(C_1)$ (BR 1)
2. $\mathrm{nev}(B_1) \to \mathrm{ev}(C_1)$, $\mathrm{nev}(B_1) \to \mathrm{nev}(C_1) \models \bot$ (BR 11)
3. $\mathrm{alw}(B_1) \to \mathrm{alw}(C_1) \models \mathrm{alw}(B_1) \to \mathrm{ev}(C_1)$ (BR 2)
4. $\mathrm{alw}(B_1) \to \mathrm{ev}(C_1)$, $\mathrm{alw}(B_1) \to \mathrm{nev}(C_1) \models \bot$ (BR 14)
5. $\mathrm{ev}(B_1) \to \mathrm{nev}(C_1)$, $\mathrm{nev}(B_1) \to \mathrm{nev}(C_1) \models B^{in} \to C_0^{out}$, $B_0^{out} \to C_0^{out}$, $B_1^{out} \to C_0^{out}$
6. $\mathrm{ev}(B_1) \to \mathrm{nev}(C_1)$, $\mathrm{alw}(B_1) \to \mathrm{nev}(C_1) \models B_1^{out} \to C_0^{out}$

[3] Thus the attribute set of a counterexample must be a concept intent in the final test context.

7. $\mathrm{ev}(B_1) \to \mathrm{nev}(C_1)$, $\mathrm{alw}(B_1) \to \mathrm{ev}(C_1) \models \bot$
8. $\mathrm{ev}(B_1) \to \mathrm{alw}(C_1) \models \mathrm{ev}(B_1) \to \mathrm{ev}(C_1)$, $\mathrm{alw}(B_1)) \to \mathrm{ev}(C_1)$, $\mathrm{alw}(B_1)) \to \mathrm{alw}(C_1)$ (BR 3)
9. $\mathrm{ev}(B_1) \to \mathrm{ev}(C_1) \models \mathrm{alw}(B_1) \to \mathrm{ev}(C_1)$ (BR 4)
10. $\mathrm{ev}(B_1) \to \mathrm{ev}(C_1)$, $\mathrm{nev}(B_1) \to \mathrm{ev}(C_1) \models B^{in} \to C_1^{out}$, $B_0^{out} \to C_1^{out}$, $B_1^{out} \to C_1^{out}$, $\mathrm{ev}(B_1) \to \mathrm{alw}(C_1)$
11. $B_1^{out} \to C_1^{out} \models \mathrm{ev}(B_1) \to \mathrm{ev}(C_1)$, $\mathrm{alw}(B_1) \to \mathrm{ev}(C_1)$, $\mathrm{alw}(B_1) \to \mathrm{alw}(C_1)$ (BR 4, BR 5 $\Leftarrow$ Proposition 1(4)(5))
12. $B_1^{out} \to C_0^{out} \models \mathrm{alw}(B_1) \to \mathrm{nev}(C_1)$ (BR 9 $\Leftarrow$ Proposition 1(4))
13. $B_0^{out} \to C_1^{out} \models \mathrm{nev}(B_1) \to \mathrm{ev}(C_1)$, $\mathrm{nev}(B_1) \to \mathrm{alw}(C_1)$ (BR 1, 10 $\Leftarrow$ Proposition 1(4))
14. $B_0^{out} \to C_0^{out} \models \mathrm{nev}(B_1) \to \mathrm{nev}(C_1)$ (BR 6 $\Leftarrow$ Proposition 1(4))

It remains to prove the rules of this stem base, which is easy; we are giving some hints: BR 1, 2, 3, and 4 are based on $\mathrm{alw}(A) \to \mathrm{ev}(A)$, $A \subseteq M$, and BR 11 and 14 on $\mathrm{nev}(C_1)$, $\mathrm{ev}(C_1) \to \bot$ ($\bot$ = set of all attributes, and the corresponding object set is empty).

7.: Since $\mathrm{nev}(C_1)$ and $\mathrm{ev}(C_1)$ do not occur together by definition, the combination of the two implications has support 0 in the test context. In the underlying contexts, the premise $\mathrm{alw}(B_1)$ (a subcase of $\mathrm{ev}(B_1)$) is no attribute of any state. The implication $\mathrm{alw}(B_1) \to \bot$ holds, which has not been considered explicitly.

10.: Inversely, in all possible cases the states / transitions have the attribute $\mathrm{ev}(C_1)$, and therefore also $\mathrm{alw}(C_1)$ and C_1^{out}. Explicitly: $\top \to \mathrm{ev}(C_1)$, $\top \to \mathrm{alw}(C_1)$, $\top \to C_1^{out}$. 5. is a parallel rule concerning $\mathrm{nev}(C_1)$.

Rules 7. and 10. suggest that implications with empty premise $\top$ or conclusion $\bot$ should be considered explicitly. If the counterexamples have maximal attribute sets, as a conclusion it can be stated that we have derived a set of rules representing a minimal, sound and complete entailment calculus for the selected classes of implications for transition and state contexts.

3 Results: Sporulation in *Bacillus subtilis*

In order to demonstrate the characteristics of the proposed method, we will apply it to a gene regulatory network assembled in [5] and transformed to a Petri net as well as a Boolean network in [16].

B. subtilis is a gram positive soil bacterium. Under extreme environmental stress, it produces a single endospore, which can survive ultraviolet or gamma radiation, acid, hours of boiling or long periods of starvation. The bacterium leaves the vegetative growth phase in favour of a dramatically changed and highly energy consuming behaviour, and it dies at the end of the sporulation process. This corresponds to a switch between two clearly distinguished genetic programs, which are complex but comparatively well understood.

By literature and database search, de Jong et al. [5] identified 12 main regulators, constructed a model of piecewise linear differential equations and obtained realistic simulation results. An exogenous signal (starvation) triggers the phosphorylation of the transcription factor Spo0A to Spo0AP by the kinase KinA;

this process is reversible by the phosphatase Spo0E. Spo0AP is necessary to transcribe SigF, which regulates genes initiating sporulation and therefore is an output of the model. The interplay with other transcription factors AbrB, Hpr, SigA, SigF, SigH and SinR is graphically represented in [5, Figure 3]; SinI inactivates SinR by binding to it. SigA and Signal are considered as an input to the model and are always on. Table 4 lists the Boolean equations building the model in [16] (communicated by the author). They exhibit a certain degree of nondeterminism, since the functions for the *off* fluents sometimes are not the negation of the *on* functions. This accounts for incomplete or inconsistent knowledge. In the case of state transitions determined by k conflicting function pairs, we generated 2^k output states.

Table 4. Boolean rules for the nutritional stress response regulatory network, derived in [16] from [5]. $\overline{x}\hat{=}\neg x$, $x+y\hat{=}x\vee y$, $xy\hat{=}x\wedge y$.

$\overline{\text{AbrB}}$	$= \text{SigA}\ \overline{\text{AbrB}}\ \overline{\text{Spo0AP}}$
$\overline{\text{AbrB}}$	$= \overline{\text{SigA}} + \text{AbrB} + \text{Spo0AP}$
$\overline{\text{SigF}}$	$= (\text{SigH}\ \text{Spo0AP}\ \overline{\text{SinR}}) + (\text{SigH}\ \text{Spo0AP}\ \text{SinI})$
$\overline{\text{SigF}}$	$= (\text{SinR}\ \overline{\text{SinI}}) + \overline{\text{SigH}} + \overline{\text{Spo0AP}}$
$\overline{\text{KinA}}$	$= \text{SigH}\ \overline{\text{Spo0AP}}$
$\overline{\text{KinA}}$	$= \overline{\text{SigH}} + \text{Spo0AP}$
$\overline{\text{Spo0A}}$	$= (\text{SigH}\ \overline{\text{Spo0AP}}) + (\text{SigA}\ \overline{\text{Spo0AP}})$
$\overline{\text{Spo0A}}$	$= (\overline{\text{SigA}}\ \text{SinR}\ \overline{\text{SinI}}) + (\overline{\text{SigH}}\ \overline{\text{SigA}}\ ) + \text{Spo0AP}$
$\overline{\text{Spo0AP}}$	$= \text{Signal}\ \text{Spo0A}\ \overline{\text{Spo0E}}\ \text{KinA}$
$\overline{\text{Spo0AP}}$	$= \overline{\text{Signal}} + \overline{\text{Spo0A}} + \text{Spo0E} + \overline{\text{KinA}}$
$\overline{\text{Spo0E}}$	$= \text{SigA}\ \overline{\text{AbrB}}$
$\overline{\text{Spo0E}}$	$= \overline{\text{SigA}} + \text{AbrB}$
$\overline{\text{SigH}}$	$= \text{SigA}\ \overline{\text{AbrB}}$
$\overline{\text{SigH}}$	$= \overline{\text{SigA}} + \text{AbrB}$
$\overline{\text{Hpr}}$	$= \text{SigA}\ \text{AbrB}\ \overline{\text{Spo0AP}}$
$\overline{\text{Hpr}}$	$= \overline{\text{SigA}} + \overline{\text{AbrB}} + \text{Spo0AP}$
SinR	$= (\text{SigA}\ \overline{\text{AbrB}}\ \overline{\text{Hpr}}\ \overline{\text{SinR}}\ \overline{\text{SinI}}\ \text{Spo0AP}) +$
	$(\text{SigA}\ \overline{\text{AbrB}}\ \overline{\text{Hpr}}\ \text{SinR}\ \text{SinI}\ \text{Spo0AP})$
$\overline{\text{SinR}}$	$= \overline{\text{SigA}} + \text{AbrB} + \text{Hpr} + (\text{SinR}\ \overline{\text{SinI}}) + (\overline{\text{SinR}}\ \text{SinI}) + \overline{\text{Spo0AP}})$
$\overline{\text{SinI}}$	$= \text{SinR}$
$\overline{\text{SinI}}$	$= \overline{\text{SinR}}$
SigA	$=$ TRUE (input to the model)
Signal	$=$ TRUE or FALSE (constant, depending on the input state)

3.1 Simulation Starting from a State Typical for the Vegetative Growth Phase

We performed supplementary analyses of the transitions starting from a typical state without the starvation signal [16, Table 4]. The concept lattice for the resulting transitive context (Table 2, with a part of the attribute set only) is shown in Figure 1. The larger circles at the bottom represent *object concepts*;

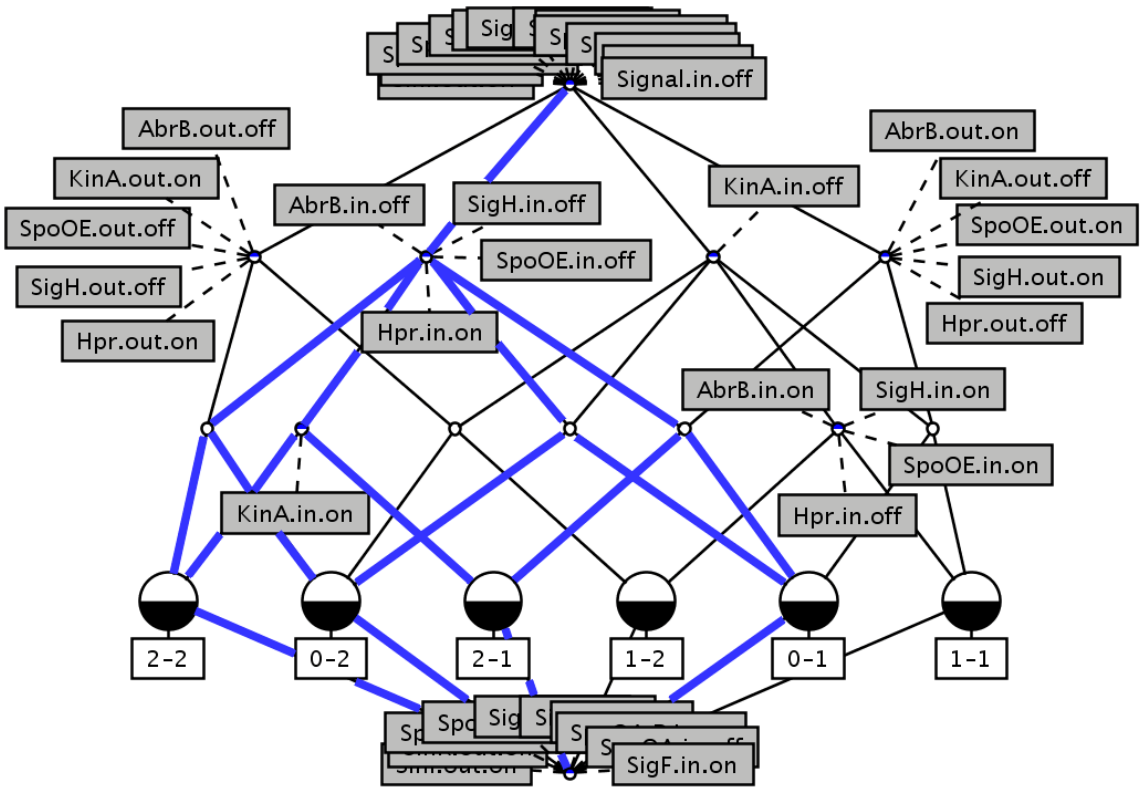


Fig. 1. Concept lattice (computed and drawn with Concept Explorer) representing a simulation without nutritional stress. *Signal*: starvation; *AbrB, Hpr, SigA, SigF, SigH, SinR, Spo0A* (phosporylated form *Spo0AP*): transcription factors; *KinA*: kinase; *Spo0E*: phosphatase; *SinI* inactivates *SinR* by binding to it. *i-j* indicates a transition $(\varphi_i^{in}, \varphi_j^{out})$. *Bold / blue lines*: Filter (superconcepts) and ideal (subconcepts) of the concept $(\{(\varphi_0^{in}, \varphi_1^{out}), (\varphi_0^{in}, \varphi_2^{out}), (\varphi_2^{in}, \varphi_1^{out}), (\varphi_2^{in}, \varphi_2^{out})\}$, {AbrB.in.off, SigH.in.off, SpoOE.in.off, Hpr.in.on}).

their extents are the four single transitions with the input state at $t = 0$ or $t = 2$, and the intents are all attributes above a concept. Thus for instance, the two latter transitions have the attribute KinA.in.on in common, designating the respective concept. Moreover, they are distinguished unambigously from other sets of transitions by this attribute – the concept is *generated* by "KinA.in.on".

Implications of the stem base can be read from the lattice. For instance there are implications between the generators of a concept:

$$< 4 > \quad \text{AbrB.in.off} \rightarrow \text{SigH.in.off, SpoOE.in.off, Hpr.in.on} \tag{7}$$

Analogous implications hold for the attributes of the conclusion, and there are implications between attributes of sub- and superconcepts. $< 4 >$ indicates that the rule is supported by four transitions.

The bottom concept has an empty extent. Its intent is the set of attributes never occuring during this small simulation. The top concept does not have an empty intent – as it is often the case -, but it consists of 10 attributes common to all 6 transitions. The corresponding rule has an empty body ($\top$):

$$< 6 > \top \rightarrow \text{Signal.in.off, SigA.in.on, SigF.in.off, Spo0A.in.on, Spo0AP.in.off, SinR.in.off, SinI.in.off, Signal.out.off, SigA.out.on, SigF.out.off, Spo0A.out.on, Spo0AP.out.off SinR.out.off SinI.out.off} \tag{8}$$

Related to the simulation in the presence of nutritional stress, the transitive context has about 20 transitions, 500 concepts and 50 implications. In a such case it is more convenient to query the implicational knowledge base. But also for the visualization of large concept hierarchies, there exist more sophisticated tools like the ToscanaJ suite [`http://sourceforge.net/projects/toscanaj/`].

3.2 Analysis of All Possible Transitions

In order to analyse the dynamics of the *B. subtilis* network exhaustively, we generated 4224 transitions from all possible $2^{12} = 4096$ initial states (thus the rules are nearly deterministic). There were 11.700 transitions in the transitive context, from which we computed the stem base containing 524 implications with support > 0, but $11.023.494 \approx 2^{24}$ concepts.

It was not feasible to provide biological evidence for a larger part of the implications, within the scope of this methodological study. This could be done by literature search, especially automatic text mining, by new specialized experiments, or - in a faster, but less reliable way - by comparison with high-throughput observed time series [18, 3.2]. Instead we will give examples for classes of implications that can be validated or falsified during attribute exploration in specific ways. We start with the examples of [16, 4.3].

- "For example, we know that in the absence of nutritional stress, sporulation should never be initiated [5]. We can use model checking to show this holds in our model by proving that no reachable state exists with SigF = 1 starting from any initial state in which Signal = 0, SigF = 0 and Spo0AP = 0." [16, 341] This is equivalent to the rule following from the stem base:

$$\text{Signal.in.off, SigF.in.off, Spo0AP.in.off} \rightarrow \text{SigF.out.off,} \tag{9}$$

- SigF.out.on → KinA.out.off, Spo0A.out.off, Hpr.out.off, AbrB.out.off: Spo0AP is reported to activate the production of SigF but also repress its own production (mutual exclusion). [5]
- SigH.out.off → AbrB.out.off, SpoOE.out.off, SinR.out.off, SinI.out.off All these genes are regulated $\overline{gene.out} = \overline{\text{SigA.in}} + \text{AbrB.in}$ (+ ...).

In our approach, such dependencies and mutual exclusions can be checked systematically. We searched the stem base for further interesting and simple implications:

$$< 4500 > \text{Spo0AP.in.on, KinA.out.off} \rightarrow \text{Hpr.out.off} \tag{10}$$

$$< 4212 > \text{SigH.in.on. KinA.out.off} \rightarrow \text{Hpr.out.off} \tag{11}$$

$$< 3972 > \text{AbrB.in.off, KinA.out.off} \rightarrow \text{Hpr.out.off} \tag{12}$$

$\overline{\text{Hpr}}$ and $\overline{\text{KinA}}$ are determined by different Boolean functions, but they are coregulated in all states emerging from any input state characterized by the single attributes Spo0AP.on, SigH.on or AbrB.on.

$$\begin{aligned} < 3904 > \text{AbrB.out.on} \rightarrow\ & \text{SigA.in.on, SigA.out.on, SigF.out.off,} \\ & \text{Spo0A.out.on, Spo0E.out.on, SigH.out.on,} \\ & \text{Hpr.out.off, SinR.out.off, SinI.out.off} \end{aligned} \tag{13}$$

AbrB is an important "marker" for the regulation of many genes, which is understandable from the Boolean rules with hindsight. By a PubMed query, a confirmation was found for downregulation of SigF together with upregulation of AbrB [17].

Finally we entered sets of interesting attributes as facts into the PROLOG knowledge base, such that a derived implication was computed.[4] Complementary to (9), we searched after conditions for the switch towards sporulation (SigF.out.on) and found the implication

$$\begin{aligned} & \text{SigF.in.off, Spo0AP.in.off, SigF.out.on} \\ \rightarrow\ & \text{Signal.in.on. Signal.out.on, SigA.in.on, SigA.out.on, Spo0AP.out.off,} \\ & \text{Spo0A.out.off, AbrB.out.off, KinA.out.off, Hpr.out.off.} \end{aligned} \tag{14}$$

The latter four attributes follow immediately from the Boolean rules, but Spo0AP.out.off depends in a more complex manner on the premises. It is also noteworthy that the class of input states developing to a state with attribute SigF.out.on is only characterized by the common attributes Signal.in.on and SigA.in.on, i.e. the initial presence or absence of no other gene is necessary for the initiation of sporulation.[5]

4 Discussion

The present work translates observations and simulations of discrete temporal transitions into the language of formal concept analysis. The application to a well studied gene regulatory network showed how a model can be validated in a systematic way, by drawing clear and complete consequences from the theory (the knowledge based network), and we found interesting new transition rules. The approach could be expanded by accounting for the change of the network structure itself in strongly different biological situations, e.g. with or without stress. Thus in ongoing work we adapt a literature based network to observed transcriptome time series, resulting in two sets of Boolean functions related to the stimulation of human fibroblast cells by the cytokines Tnfα or Tgfβ.

Until now we have established the foundation in order to exploit manyfold mathematical results of FCA for the analysis of gene expression dynamics and

[4] And accordingly the closure of the attribute set.

[5] For this complete simulation, the conditions Signal = SigA = TRUE had been dropped, but they were supposed to be constant.

of discrete temporal transitions in general. An important question is: How can attribute exploration be split into partial problems, in this special case? For instance, one could focus on a specific set of genes first, which is understandable as a *scaling* [4, Definition 28]. Then the decomposition theory of concept lattices will be useful, which permits an elegant description by means of the corresponding formal contexts [4, Chapter 4].

The price of the logical completeness is its computational complexity. In this regard the status of attribute exploration has not yet fully been clarified. Computation time strongly depends on the logical structure of the context, and there exist cases where the size of the stem base is exponential in the size of the input [10]. However, deriving an implication from the stem base is possible in linear time, related to the size of the base, and the PROLOG queries in Section 3.2 were very fast. As demonstrated in Section 2.4, attribute exploration can be shortened by background knowledge. Further it will be crucial to decide implications without the necessity to generate all possible transitions. For that purpose, model checking [6] could be a promising approach, or the structural and functional analysis of Boolean networks by an adaptation of metabolic network methods in [9]. There, determining activators or inhibitors corresponds to the kind of rules found by our method, and logical steady state analysis indicates which species can be produced from the input set and which not. An exciting direction of research would be to conclude dynamical properties of Boolean networks from their structure and the transition functions, e.g. by regarding them as polynomial dynamical systems over finite fields [12, Section 4] and by exploiting theoretical work in the context of cellular automata [12, Section 6].

The present work is a first step to use the potential of formal concept analysis for solving questions within systems biology. As indicated, many directions of research are possible. We encourage their investigation and are open to any collaboration with mathematicians, computer scientists or (systems) biologists.

Acknowledgement. The work was supported by the German Federal Ministry of Education and Research BMBF (FKZ 0313652A).

References

1. Chabrier-Rivier, N., et al.: Modeling and Querying Biomolecular Interaction Networks. Theor. Comp. Sc. 325(1), 25–44 (2004)
2. Choi, V., et al.: Using Formal Concept Analysis for Microarray Data Comparison. Advances in Bioinformatics and Computational Biology 5, 57–66 (2006)
3. Ganter, B., Stumme, G., Wille, R. (eds.): Formal Concept Analysis. LNCS (LNAI), vol. 3626. Springer, Heidelberg (2005)
4. Ganter, B., Wille, R.: Formal Concept Analysis - Mathematical Foundations. Springer, Heidelberg (1999)
5. de Jong, H., et al.: Qualitative Simulation of the Initiation of Sporulation in *Bacillus subtilis*. Bulletin of Mathematical Biology 66, 261–299 (2004)
6. Esparza, J.: Model checking using net unfoldings. Sci. Comput. Programm 23, 151–195 (1994)

7. von Hentig, H.: Magier oder Magister? Über die Einheit der Wissenschaft im Verständigungsprozess. Suhrkamp, Frankfurt (1974)
8. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, New York (1993)
9. Klamt, S., et al.: A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinformatics 7(56) (2006)
10. Kuznetsov, S.O., Obiedkov, S.A.: Counting Pseudo-intents and #P-completeness. In: Missaoui, R., Schmidt, J. (eds.) ICFCA 2006. LNCS (LNAI), vol. 3874, pp. 306–308. Springer, Heidelberg (2006)
11. King, R.D., Garrett, S.W., Coghill, G.M.: On the Use of Qualitative Reasoning to Simulate and Identify Metabolic Pathways. Bioinformatics 21(9), 2017–2026 (2005)
12. Laubenbacher, R.: Algebraic Models in Systems Biology. In: Anai, H., Horimoto, K. (eds.) Algebraic Biology 2005, pp. 33–35. Universal Academy Press, Tokyo (2005)
13. Motameny, S., Versmold, B., Schmutzler, R.: Formal Concept Analysis for the Identification of Combinatorial Biomarkers in Breast Cancer. In: Medina, R., Obiedkov, S.A. (eds.) ICFCA 2008. LNCS, vol. 4933, pp. 229–240. Springer, Heidelberg (2008)
14. Peirce, C.S.: How to Make Our Ideas Clear. In: Hartshorne, C., Weiss, P. (eds.) Collected papers. Harvard University Press, Cambridge/Mass (1931-1935)
15. Ganter, B., Rudolph, S.: Formal Concept Analysis Methods for Dynamic Conceptual Graphs. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 143–156. Springer, Heidelberg (2001)
16. Steggles, L.J., et al.: Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach. Bioinformatics 23(3), 336–343 (2007)
17. Tomas, C.A., et al.: DNA array-based transcriptional analysis of asporogenous, nonsolventogenic Clostridium acetobutylicum strains SKO1 and M5. J. Bacteriol. 15, 4539–4547 (2003)
18. Wollbold, J.: Attribute Exploration of Discrete Temporal Transitions. In: Gély, A., et al. (eds.) ICFCA 2007, Clermont-Ferrand, pp. 121–130 (2007)
19. Wollbold, J., Huber, R., Wolff, K.E.: Conceptual Representation of Gene Expression Processes. In: Knowledge Processing in Practice 2007. LNCS (LNAI). Springer, Heidelberg (to appear, 2008)
20. Zickwolff, M.: Rule Exploration: First Order Logic in Formal Concept Analysis. PhD thesis. University of Technology, Darmstadt (1991)

Author Index