

# Cycles and Communicating Classes in Membrane Systems and Molecular Dynamics

Michael Muskulus<sup>a</sup>, Daniela Besozzi<sup>b</sup>, Robert Brijder<sup>c</sup>  
Paolo Cazzaniga<sup>d</sup>, Sanne Houweling<sup>e</sup>, Dario Pescini<sup>d</sup> and  
Grzegorz Rozenberg<sup>c</sup>

<sup>a</sup>*Leiden University, Mathematical Institute, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands*

*Corresponding author. E-mail: muskulus@math.leidenuniv.nl*

<sup>b</sup>*Università degli Studi di Milano, Dipartimento di Informatica e Comunicazione, Via Comelico 39, 20135 Milano, Italy*

<sup>c</sup>*Leiden University, Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands*

<sup>d</sup>*Università degli Studi di Milano-Bicocca, Dipartimento di Informatica, Sistemistica e Comunicazione, Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy*

<sup>e</sup>*Vrije Universiteit Amsterdam, Faculteit der Bewegingswetenschappen, Van der Boechorststraat 9, 1081 BT Amsterdam, The Netherlands*

---

## Abstract

We are considering sequential membrane systems and molecular dynamics from the viewpoint of Markov chain theory. The configuration space of these systems (including the transitions) is a special kind of directed graph, called pseudo-lattice digraph, which is closely related to the stoichiometric matrix. Taking advantage of the monoidal structure of this space, we introduce the algebraic notion of precycle. A precycle leads to the identification of cycles by means of the concept of defect, which is a set of geometric constraints on configuration space. Two efficient algorithms to evaluate precycles and defects are given: one is an algorithm due to Contejean and Devie, the other is a novel branch-and-bound tree search procedure. Cycles partition configuration space into equivalence classes, called the communicating classes. The structure of the communicating classes in the free regime — where all rules are enabled — is analysed: testing for communication can be done efficiently. We show how to apply these ideas to a biological regulatory system.

DOI of the journal version: 10.1016/j.tcs.2006.11.027

*Key words:* membrane systems, communicating classes, molecular dynamics, cycles in digraphs, vector addition systems.

*PACS:* 02.10.Ox, 82.39.Rt, 89.75.-k

---

## 1 Introduction

In the framework of Molecular Dynamics [12] (MD), chemical reactions in a well-mixed volume are simulated under simplifying assumptions. The probability distributions of collisions with ensuing reactions between chemical objects can be calculated from empirically measured reaction constants, and through stochastic simulation the time evolution of the system and its behaviour can be studied. The standard algorithm used is the Stochastic Simulation Algorithm (SSA) due to Gillespie [12]. Recently, important improvements on the simulation speed have been gained through the use of approximate versions of the SSA, in particular the Poisson or binomial  $\tau$ -leap methods [13,34] and the efficient implementation of the reaction rate update process [11]. Yet all these methods suffer from their forbiddingly large time demands under realistic conditions, i.e. large numbers of molecules (whereas the number of different chemical species encountered is usually quite small).

Independently from this research, the framework of membrane systems [23,24], for short P systems, has emerged as a well-studied model in the natural computing community. It features parallel, non-deterministic multiset rewriting in hierarchical reaction volumes (membranes). A special class are so-called Dynamical Probabilistic P systems [26] (DPP) which introduce probabilities on rule applications, and are increasingly being used to simulate and understand biological processes. A single-membrane DPP which runs sequentially (1sDPP), choosing one rewriting rule at each evolution step according to some probability law, is equivalent to a non-deterministic version of a corresponding MD system. Running a stochastic simulation on a 1sDPP, the resulting trajectory is the same as for a corresponding MD system, apart from the fact that there is ignorance about the reaction times. In particular, simulation time is still the main problematic issue for realistic applications.

Many efforts in the membrane system community are devoted to modelling and simulating biological systems, but there is relatively few research on analytic properties of such systems. Chemical Reaction Network Theory (CRNT) [15] deals with corresponding continuous systems of ordinary differential equations and is able to make strong statements about dynamic features, e.g. fixed points. Although there are some attempts [29,2], a similar theory for the discrete case has yet to be developed (see [22] for further possibilities and other issues of relevance).

Here we are studying 1sDPP systems in the context of Markov chain theory. The first step will be to understand the structure of the state space in terms of communicating classes (and a full probabilistic treatment could then be undertaken in the future). This characterization leads to a partition of configuration space and makes it possible to recognize dynamical patterns of the analysed systems without performing simulations. For generic Markov chains the communicating classes can be computed by generating the state set and checking for mutual reachability, i.e.

cycles in the evolution digraph. Algorithms exist for both issues, but tend to focus mainly on storage efficiency [19] and on reachability issues [4,18]. In this generality, the known algorithms are relatively inefficient. In this paper we take into account the monoidal structure of the rewriting rules to derive more efficient and adapted algorithms.

Two problems arise when the system is observed near the origin of configuration space: the first one concerns the possibility to find the minimal configurations that allow to apply a given sequence of rules. The second one concerns the possibility to find the minimal configurations that allow to apply a set of unordered rules. Both questions are answered by introducing the concept of *defect* in Section 3.7. We give a novel branch-and-bound tree search algorithm to efficiently calculate defects in Section 3.8.

All the topological information of a membrane system is contained in the stoichiometric matrix. This information is exploited to find the recurring rule sequences, i.e. the cycles, of the system. This is done by applying an incremental algorithm due to Contejean and Devie [6] with the stoichiometric matrix as input, to find the *pre-cycles*, i.e. multisets of rules which lead to cycles when applicable. The precycles are then used as input to the tree search algorithm which calculates their defects, from which we can identify the cycles. This scheme is particularly interesting for systems involving a small number of different kinds of objects, a common case in Systems Biology.

It is also possible to consider the system far from the origin in the so-called *free* regime. This simplifies the analysis, since in this regime defects play no role and all rules are always applicable. Section 4 aims to answer the question of how many communicating classes exist in the free regime. It is shown in Section 4.2 that testing for communication can be done efficiently. Finally, some of the ideas developed are tested in Section 5 on a biological regulatory network, the LacZ-LacY operon.

## 2 Background

General knowledge of molecular dynamics [12] and membrane systems [24] will be necessary for understanding the paper. Therefore we give a short review of these topics in Section 2.1 and 2.2, respectively. The special case of Dynamical Probabilistic P systems is addressed in Section 2.3. In some of the examples we use a graphical representation by place-transition Petri nets [27,20] for illustration. Relevant notions from graph theory and Markov chain theory are reviewed in Section 2.4 and are based on the monographs [8,1] and [3], respectively.

At a few points we will use basic notions from convex analysis and linear programming, which can be gleaned from [32], if necessary. Readers unfamiliar with partial

orders might find [7] another useful reference.

## 2.1 *Molecular dynamics*

The standard approach to simulating a chemical system is due to Gillespie [12] who derived the relevant methodology and algorithms. This model is based on the following assumptions: (i) a well-mixed single reaction volume, populated with spherical objects, (ii) instantaneous chemical reactions via collisions of particles. Reaction rates have to be given as parameters, e.g. from empirical measurements or theoretical arguments, with underlying exponential waiting time distributions. Other approaches include Green's function reaction dynamics [36] and analytical treatment based on the master equation [35].

## 2.2 *Membrane systems*

Membrane systems [23] (P systems, for short) are a formal computational model for biochemical reactions in several hierarchically connected reaction volumes. The structural part of a P system consists of a nested *membrane structure*, formally represented (usually) as a tree. To each membrane corresponds its (inner) *region*, which can contain other membranes and a multiset of objects, i.e. chemical species, which can undergo reactions. The outermost membrane, the root in the tree representation, is called the *skin membrane*, since it separates the system from the *environment*, which is an assumed outer region in which the system resides.

The dynamical part is given by rewriting rules on the objects, corresponding to possible biochemical reactions, which are associated with a certain region. Since we are dealing with multisets, it is important to differentiate between *object species* (for short *species*), which are the different chemical species modelled by the system, and *object representatives* (for short *objects*); such that different objects can belong to the same species.

Since rules are associated with regions, objects could undergo potentially different reactions in different regions. In the standard mode of behaviour, the rewriting rules are applied in a *maximal parallel* fashion, i.e. a global clock is assumed, and at each time step a set of applicable rules is chosen nondeterministically for each region, using up the available objects, such that no more rules can be added to the applicable set of rules at that time step. This can be thought of as modelling parallel biochemical reaction channels, and is a powerful computational feature (as well as a convenience for analysis). The products of the reactions will be available without delay in the following time step for new reactions. Note also that a rule can specify a possible transportation of product objects by one step in the membrane hierarchy, i.e. each product can move up to the parent region, go down into one of the child

membrane regions (given they exist), or stay put. Formal details can be found in the monograph [24].

The *maximal parallel* execution of rules has led to some controversy [21]. It facilitates analysis, but seems unnatural with respect to biochemical reality. In fact, simulations in molecular dynamics are done sequentially for good reasons<sup>1</sup> and apart from very specialized applications, e.g. in periodically induced chemical reactions, maximal parallelism has to be considered an unwanted artifact in most biochemical applications.

### 2.3 *Dynamic probabilistic P systems*

The nondeterminism in the definition of membrane systems has to be replaced with some probability law, if we are to consider statistical properties of such a system, and not only topological ones. Although we do not consider these issues in this paper, we give some background information here, since this defines our long-term motivation and leaves room for future research.

The recently introduced *Dynamical Probabilistic P Systems* [26,25] modify the definition of membrane systems by introducing a rate constant  $k \in \mathbb{R}^+$  to each rule, which dynamically determines a probability law. More precisely, this is done by considering a combinatorial factor<sup>2</sup> for all possible choices of the needed objects from the available ones — as prescribed by mass-action kinetics — multiplying this by the rate constant, normalizing the resulting numbers to probabilities and then randomly choosing reactions according to this probability law. The details can be found in [26,25] but are not needed for our presentation here.

Note: A similar approach based on arbitrary reaction maps, i.e. not restricted to mass-action kinetics, has been developed independently and is mentioned here for the interested reader. Its key element is the so-called *metabolic algorithm* [2].

---

<sup>1</sup> It can be argued that nature works sequentially almost surely (in the sense of probability theory, i.e. with respect to a set of exceptions with measure zero). When considering kinetic particle models, for example, the probability of two collisions taking place at the same time is zero. The same applies to a quantum-mechanical treatment (cf. [16]). If reactions are considered not to be instantaneous, but take a finite time (as in the binding of reactants to a reaction complex, for example, or in the decay of metastable states), reactions can of course occur in parallel. The important point is, though, that the *initiation* of reactions has to be considered in a sequential manner, and not in a synchronized way as exemplified by maximal parallelism.

<sup>2</sup> Note: In a quantum mechanical application of membrane systems (see [16] for a possible approach) this factor equals one, since quantum mechanically different copies of the same object *cannot* be distinguished.

## 2.4 Markov chains and digraphs

Molecular dynamics and membrane systems can be studied in the more general setting of Markov chain theory [3], where (sequential) membrane systems correspond to topological Markov chains, and sDPP systems and MD systems correspond to Markov chains.

A (generalized) matrix  $P_{ij}$  with entries in  $\mathbb{R}$ ,  $i \in I, j \in J$ , for some countable index sets  $I, J$ , is a function  $I \times J \rightarrow \mathbb{R}$ . It is *stochastic*, if its row sums  $\sum_j P_{ij}$  are equal to unity for all  $i \in I$ . A *Markov chain* on a countable set (space)  $E$  is a (generalized) *stochastic matrix*  $P_{ij}$ ,  $i, j \in E$ , with nonnegative entries.

In this paper we are mainly concerned with the discrete space  $(\mathbb{N}_0^m, +)$ , for some  $m \in \mathbb{N}$ , called *configuration space* throughout, except for Section 4, where this will be  $(\mathbb{Z}^m, +)$ . Elements of configuration space are called *configurations* or *states*. The *topological Markov chain* of a Markov chain  $P_{ij}$  is the (generalized) matrix  $F_{ij}$  containing a one everywhere where  $P_{ij} > 0$ , and a zero where  $P_{ij} = 0$ , thereby encoding the information on the possible transitions of a state, without their probabilities, i.e. only on the topology of the system's behaviour. A topological Markov chain  $F_{ij}$  on configuration space corresponds to a function  $F_1$  from configuration space to subsets of itself, via

$$F_1 : \mathbb{N}_0^m \rightarrow 2^{\mathbb{N}_0^m}, \quad F_1(c) = \bigcup_{F_{cc'}=1} \{c'\} \quad (1)$$

This map is denoted the *coevolution map* of  $F_{ij}$ . It assigns to each configuration  $c \in \mathbb{N}_0^m$  its possible 1-step evolutions  $F_1(c)$  and can be extended to a set-valued map

$$F_1 : 2^{\mathbb{N}_0^m} \rightarrow 2^{\mathbb{N}_0^m}, \quad F_1(C) = \bigcup_{c \in C} F_1(c). \quad (2)$$

Its iterates  $F_n(c)$  are defined inductively via  $F_{n+1}(c) = F_1(F_n(c))$ . We also define  $F_0(c) = c$ .

A state  $c' \in \mathbb{N}_0^m$  is *reachable* from a state  $c \in \mathbb{N}_0^m$  if there exists a nonnegative integer  $k$ , such that  $c' \in F_k(c)$ . We denote reachability by  $c \rightarrow c'$ . The *evolution digraph* of the topological Markov chain  $F_{ij}$  on configuration space  $\mathbb{N}_0^m$  is the digraph (directed graph)  $D = (V, A)$ , with vertex set  $V = \mathbb{N}_0^m$ , and arc set  $A = \{(v, v') \mid v, v' \in V, v' \in F_1(v)\}$ , and is usually an infinite<sup>3</sup> graph.

A *walk* of size  $k \in \mathbb{N}_0$  in a digraph  $D = (V, A)$  is a sequence of vertices  $(c_0, \dots, c_k)$ , such that  $(c_i, c_{i+1}) \in A$  for all  $i < k$ . We also say that the walk is *from*  $c_0$  *to*  $c_k$ . A *path* is a walk where all vertices are distinct; consequently all arcs are distinct. In

<sup>3</sup> The properties of graphs we use and the theorems we state are valid for infinite graphs, too, unless explicitly remarked. See [8] for an introduction into infinite graph theory.

a digraph, a walk  $c = (c_0, \dots, c_k)$  forms a *cycle*, if  $c_0 = c_k$ ,  $k > 0$ . If the vertices  $c_0, \dots, c_{k-1}$  of the cycle are distinct, we speak of a *simple* cycle. Note that we deviate from the usual definition where cycles are always considered to be simple.

Clearly,  $c'$  is reachable from  $c$  iff there is a walk from  $c$  to  $c'$  in the evolution digraph. Two states  $c, c' \in \mathbb{N}_0^m$  are *communicating*, if both  $c \rightarrow c'$  and  $c' \rightarrow c$ . We write this as  $c \sim c'$ . This is equivalent to the existence of a cycle which contains both  $c$  and  $c'$  (see below). The communicating relation is an equivalence relation, giving rise to a partition of configuration space into *communicating classes*, which by definition are the *strongly connected components* of the digraph. The *trivial* communicating classes, consisting of only one element, are referred to in their totality as the *transient set*  $\mathbf{Trans} \subseteq \mathbb{N}_0^m$ . Two or more communicating classes are said to be *independent* of each other if no element of one such class is reachable by any element of the others, and vice versa. A digraph is said to be *strong*, if it has only one communicating class.

We can characterize communicating classes by the cycles of the evolution digraph:

**Proposition 1** *A cycle  $c = (c_0, \dots, c_k)$  of length  $k$  in the evolution digraph lies in exactly one communicating class  $C \subseteq \mathbb{N}_0^m$ , i.e. with  $c_i \in C$  for all  $0 \leq i \leq k$ . Vice versa, each nontrivial communicating class  $C$  contains at least one nontrivial simple cycle  $c$ .*

We call the communicating class  $C$  in the above proposition the communicating class of the cycle  $c$  and denote this by  $[c]$ .

Given two transient configurations  $c, c' \in \mathbf{Trans}$ , they are *transient-equivalent* if one can be reached from the other in the transient set, i.e. by a walk that does not leave  $\mathbf{Trans}$ . We denote this by  $c \sim_t c'$ . Denote by  $(\mathbb{N}_0^m)_\sim$  the quotient of configuration space under the two equivalences above, called *reduced configuration space*, such that  $(\mathbb{N}_0^m)_\sim = (\mathbb{N}_0^m, +) / \sim / \sim_t$ . The nontrivial communicating classes are reduced to single points and the transient set is reduced to single entry/exit points, such that  $(\mathbb{N}_0^m)_\sim$  has the structure of a directed acyclic graph (DAG). Giving an initial configuration  $c_0$ , the set of all reachable states from  $c_0$  in  $(\mathbb{N}_0^m)_\sim$  forms a subdigraph of the reduced configuration space, the so-called *reduced evolution digraph*, with root  $c_0$ . This digraph is infinite, if the number of communicating classes reachable from  $c_0$  is infinite. Its leafs are the communicating classes in which the system eventually ends, i.e. where its long-term behaviour<sup>4</sup> takes place. If there is more than one leaf, one has different disconnected long-term dynamical regimes, a situation usually considered unrealistic. Note that walks in this graph need not be unique, i.e. a certain communicating class can be reached from the same initial condition via walks through different transients and/or communicating classes.

---

<sup>4</sup> There is a further subtlety involved, since the transient set could be potentially infinite. An exact study of these issues requires the use of probabilities, though.

We recall the following fact about digraphs [1], which guarantees the existence of at least one leaf and root:

**Proposition 2** *Every nonempty, acyclic digraph has a vertex of in-degree zero as well as a vertex of out-degree zero.*

### 3 Reaction cycles

In Section 3.1 we formalize the type of system we are studying. Section 3.2 introduces the stoichiometric matrix and states the two main problems of this paper. Sections 3.3 and 3.4 are a slight detour in which we discuss the particular aspects of open systems and the interesting idea of considering left- and right-hand sides of rules as single objects, so-called compounds. The algebraic notion of precycle is introduced in Section 3.5. A precycle gives rise to a set of pseudo-cycles at each point of configuration space. These are cycles, when they are applicable. The notion of defect, introduced in Section 3.7, allows to decide when this is the case. The algorithm of Contejean and Devie is explained in Section 3.6, since our algorithm for the calculation of the defects in Section 3.8 is based on it. Finally, Section 3.9 discusses mass-conservation in membrane systems, since it is interesting theoretically and the algorithms discussed before can be used to check for it in the special case of rational masses.

#### 3.1 Systems, traces and application vectors

We begin the study of sDPP systems by simplifying the notation. It is easy to show that (sequential<sup>5</sup>) multi-membrane DPP systems are equivalent to (sequential) one membrane DPP systems with each object species being replaced by a number of species, one for each membrane region in the original system [21]. To be more precise: There exists a bijection from one sDPP system  $\Pi$  to a 1sDPP system  $\Pi'$  such that the objects of  $\Pi$  are mapped bijectively to the objects of  $\Pi'$ , the rules of  $\Pi$  are mapped bijectively to the rules of  $\Pi'$  (and such that this is compatible with the mapping on the objects), and the (initial) multisets of objects  $M_0, M_1, \dots, M_{n-1}$  are mapped to a corresponding multiset  $M'_0$ .

We therefore exclusively study 1sDPP systems, which can be seen as normal forms of sDPP systems<sup>6</sup>. This has the advantage that we have to deal with rewriting rules

---

<sup>5</sup> By sequential we mean that at each time step only one applicable rule is chosen non-deterministically.

<sup>6</sup> Note: In the original definition of membrane systems, it is also possible for rules to dynamically dissolve a membrane. But in the context of DPP systems we only consider the *static* case where this is not allowed.

only, and no unnecessary complications due to transportation of the products is encountered. Furthermore, all finite sets of  $n$  elements ( $n$ -sets, for short) encountered in the definition will be canonically identified with lower intervals  $\{1, \dots, n\}$  of the non-negative integers  $\mathbb{N}$ . Thus, a multiset  $c : \mathcal{O} \rightarrow \mathbb{N}_0$  over a  $m$ -set  $\mathcal{O}$  will be identified as a vector  $c \in \mathbb{N}_0^m$  instead. By an abuse of language we still talk of multisets, though. For convenience we will also use canonical symbols  $A, B, C, \dots$ , in place of object species  $1, 2, 3, \dots$ , where we see fit.

In stating the rewriting rules of the system, we allow three different notations, depending on which is more convenient. A rule  $r$  is specified either as a tuple of multisets  $r = (q, p)$ ,  $q, p \in \mathbb{N}_0^m$  or via an arrow notation as in  $r : q \rightarrow p$ . Furthermore, in the latter case the multisets will be written out additively, such that a multiset  $(2, 0, 1)$  is written as  $2A + C$ , for example. A third type of notation as a difference vector is introduced below.

We do not allow *catalytic* rules of the form  $M_A + C \rightarrow M_B + C$ , where  $M_A, M_B$  are multisets over  $\mathcal{O}$ , and  $C \in \mathcal{O}$ . Rules of this form, i.e. where one or more objects appear simultaneously on both sides of the rule, will also be called *degenerate*. Enforcing nondegeneracy is no restriction: Since we are only interested in topological properties, i.e. in the reachability of states, we can accommodate for degenerate rules by using two rules and an intermediary state (a new object species), which seems the more realistic behaviour in biochemistry anyway, corresponding to (1) a binding action of reactants into a so-called *complex*, and then (2) the chemical reaction and dissociation of the complex into the products.

**Example 3** *Instead of the (invalid) rule  $A + C \rightarrow B + C$  we would consider the two rules  $A + C \rightarrow D$  and  $D \rightarrow B + C$ , where  $D$  is a new object species that does not occur in any other rule.*

**Lemma 4** *A rule  $r_i = (q_i, p_i)$ , with  $q_i, p_i \in \mathbb{N}_0^m$  is nondegenerate iff  $\langle p_i, q_i \rangle = 0$ , where  $\langle p_i, q_i \rangle = p_{i,1} \cdot q_{i,1} + \dots + p_{i,m} \cdot q_{i,m}$  is the standard scalar product in  $\mathbb{N}_0^m$ .*

**PROOF.** Since all entries in  $q_i$  and  $p_i$  are nonnegative, the scalar product will only be zero if  $q_{i,j}$  and  $p_{i,j}$  are not both positive for all  $1 \leq j \leq m$ . This means that no object will appear on both sides of a rule, proving necessity. Sufficiency is trivial.  $\square$

The previous lemma allows the specification of a rule as a *difference vector*  $r = p - q \in \mathbb{Z}^m$ , since we can recover the multisets  $p, q \in \mathbb{N}_0^m$  from their vector difference via:

$$p = \max(0, r) \tag{3}$$

$$q = -\min(0, r) \tag{4}$$

where  $0 \in \mathbb{Z}^m$  is the null vector, and the maxima and minima are taken component-wise, i.e.  $\max(0, r) = (\max(0, r_i))_{1 \leq i \leq m}$ , for example. In principle, there could be

confusion between this notation and the writing of  $r$  as a tuple of multisets. It will be clear from the context though, what is meant.

**Definition 5** A topological membrane system (*for short* TM system)

$$\Pi = (m, \mathcal{R}, c_0)$$

of type  $(m, n)$  consists of

- a positive integer  $m$ ,
- a  $n$ -tuple of distinct, nondegenerate rewriting rules  $\mathcal{R} = (r_1, \dots, r_n)$  of the form  $r_i \in \mathbb{Z}^m$ ,
- and  $c_0 \in \mathbb{N}_0^m$ .

In the following, a reference to  $m, n$  always refers to the above constants of a particular TM system  $\Pi$  under consideration.

A TM system  $\Pi$  corresponds to a topological Markov chain  $F_{ij}$  on configuration space, where  $F_{ij} = 1$  if  $j$  is reachable from configuration  $i$  in a single evolution step by some applicable rule, and  $F_{ij} = 0$  otherwise. This identification allows to use the tools of Markov chain theory for TM systems.

Note: A TM system is the same as an *asynchronous multiset rewriting system*, introduced in [21], except that in the latter one also allows inhibitory rules. In a TM system we are also given the rules in the form of a tuple, instead of a set, such that there is an implicit ordering of rules. This is only a notational convenience allowing for easy statement of sequences of rules. Therefore the reader should keep in mind that we tacitly assume that all properties of a TM system do not depend on this ordering.

The integer  $m$  specifies the number of different object species of the TM system  $\Pi$ , and the multiset  $c_0 \in \mathbb{N}_0^m$  is interpreted as the *initial configuration* of  $\Pi$ . The system evolves in discrete time by sequential application of rules, and we denote by  $c_t$ ,  $t = 0, 1, 2, \dots$ , the *configuration* of a TM system at time  $t$ , i.e. the multiset of objects in the system at that time. The *configuration space* of  $\Pi$  is the monoid  $(\mathbb{N}_0^m, +)$ , where addition is the usual vector addition. In fact, such a system is a kind of *vector addition system* [28], with an initial non-negative vector affixed to it and the restriction that the rules are nondegenerate.

The application of a rule  $r = p - q$  is interpreted as the removal of the *reactants*  $q \in \mathbb{N}_0^m$  from  $\Pi$ , followed by the appearance of the *products*  $p \in \mathbb{N}_0^m$ , and corresponds to the translation of a configuration  $c_t \in \mathbb{N}_0^m$  by the difference vector  $r$ , i.e.  $c_{t+1} = c_t + r$ .

Starting from an initial configuration  $c_0 \in \mathbb{N}_0^m$ , a TM system evolves by sequentially applying reaction rules. These can only be applied if there are enough reactants for each rule to be consumed:

**Definition 6** A rule  $r \in \mathbb{Z}^m$  is enabled in the state  $c \in \mathbb{N}_0^m$  if  $r + c \in \mathbb{N}_0^m$ .

This condition is equivalent to  $-\min(0, r) \leq c$ , where  $\leq$  refers to the usual (inclusion) ordering of multisets.

The following definitions are used to characterize the transitions between two configurations, keeping count of the rules that are applied in sequence. In fact, it might be possible to move from one configuration to the other just knowing what rules and how many time each of them is applied, without the need to specify a particular order. This leads to the definition of an application vector, which will be used in Section 3.5 for the definition of precycles.

**Definition 7** A trace sequence (trace, for short) in  $\Pi$  is a finite sequence  $\tau = (\tau_1, \dots, \tau_k)$ ,  $\tau_i \in \{1, 2, \dots, n\}$ .

A trace<sup>7</sup>  $\tau$  of length  $k$  will be called a  $k$ -trace for short. It corresponds to a sequence of rules  $r(\tau) = (r_{\tau_1}, \dots, r_{\tau_k})$ , where we use the implicitly given ordering of the rules.

**Definition 8** A  $k$ -trace  $\tau = (\tau_1, \dots, \tau_k)$  is enabled for a configuration  $c_0 \in \mathbb{N}_0^m$  if  $r_{\tau_1}$  is enabled in  $c_0$  and each rule  $r_{\tau_i}$ ,  $1 < i \leq k$ , is enabled in  $c_{i-1} = c_{i-2} + r_{\tau_{i-1}}$ .

In order to calculate the cumulative effect of a trace on the TM system, the following notion is useful:

**Definition 9** Given a  $k$ -trace  $\tau$ , the corresponding cumulative vector trace is  $v(\tau) = (v_0, v_1, \dots, v_k)$ , where  $v_0 = 0$ ,  $v_j = \sum_{i=1}^j e_{\tau_i} \in \mathbb{N}_0^n$  and the canonical unit vectors  $e_j$  are zero everywhere, except at the  $j$ -th coordinate, which is one.

Note that there is a one-to-one correspondence between traces  $\tau$  and their cumulative vector traces  $v(\tau)$ . Therefore, we define a cumulative vector trace to be enabled, iff its trace is enabled. We call  $x \in \mathbb{N}_0^n$  an *application vector* in this context and  $(v(\tau))_k \in \mathbb{N}_0^n$  the *application vector of the  $k$ -trace  $\tau$* . The latter is denoted by  $|\tau|$ , for short, and counts how often each rule has been applied in the  $k$  time steps that have been traced by  $\tau$ .

**Definition 10** The trace set of an application vector  $x \in \mathbb{N}_0^n$  is  $\Xi(x) = \{\tau \mid |\tau| = x\}$ .

The trace set consists of all permutations of the elements of its application vector, which is a multiset of rules. Its elements are therefore called multiset orderings. The following lemma gives a formal description:

**Lemma 11** Given a multiset  $x \in \mathbb{N}_0^n$  of size  $\|x\| = \sum_i x_i$ , a multiset ordering  $\sigma \in \Xi(x)$

---

<sup>7</sup> In computer science this is often called a *control sequence*, but we prefer the name trace, since we usually observe the system's behaviour (in applications to biochemistry) instead of prescribing it.

of  $x$  is a sequence  $(\sigma_1, \dots, \sigma_{\|x\|})$ , with  $\sigma_i \in \{1, \dots, n\}$ ,  $0 \leq i \leq \|x\|$ , such that

$$\sum_{1 \leq i \leq n} \delta_{\sigma_i}^j = x_j \quad (5)$$

for all  $j \leq n$ , where  $\delta_{\sigma_i}^j$  denotes the Kronecker delta, which is zero if  $\sigma_i \neq j$ , and has value one otherwise.

In particular,  $\tau$  is a multiset ordering of  $|\tau|$ , and a  $\|\tau\|$ -trace<sup>8</sup>. The trace set contains

$$|\Xi(x)| = \binom{\|x\|}{|x|_1, |x|_2, \dots, |x|_n} = \frac{\|x\|!}{|x|_1! \cdot |x|_2! \cdots |x|_n!} \quad (6)$$

multiset orderings in total.

**Example 12** Given  $x = (2, 1, 2)^t$  we get

$$\Xi(x) = \{(1, 1, 2, 3, 3)^t, (1, 2, 1, 3, 3)^t, \dots\},$$

such that  $|\Xi(x)| = \binom{5}{1, 2, 2} = 30$ .

### 3.2 The stoichiometric matrix

The notion of stoichiometric matrix [9,29] is given in the context of membrane systems as follows:

**Definition 13** The stoichiometric matrix of a TM system  $\Pi$  is

$$R = \begin{pmatrix} r_1 & r_2 & \cdots & r_n \end{pmatrix} \quad (7)$$

This is a  $(m, n)$ -matrix of integer entries, such that the  $i$ -th rule  $r_i$  corresponds to the  $i$ -th column, and the  $j$ -th row corresponds to the possible changes in object species  $j$  by all  $n$  rules. Note again: The stoichiometric matrix depends on the implicit ordering of the rules, but we are only interested in properties which are invariant with respect to the ordering.

Since the stoichiometric matrix contains all the topological information about a TM system, it can be used to define all other quantities of interest.

**Example 14** Consider  $\Pi_B = (m, \mathcal{R}, c_0)$ , where  $m = 3$ , and  $\mathcal{R} = (r_1, \dots, r_4)$  is given by  $r_1 : \lambda \rightarrow A$ ,  $r_2 : A \rightarrow B$ ,  $r_3 : 2A + B \rightarrow C$ ,  $r_4 : C \rightarrow 3A$ , where  $\lambda$  designates the empty multiset (and can be interpreted as the effect of the environment, modelling

<sup>8</sup> Here the norm  $\|\cdot\|$  indicates that we take the modulus  $|\cdot|$  of the application vector  $|\tau|$ .

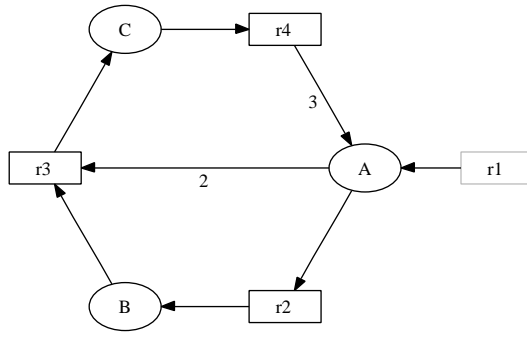


Figure 1. Petri net representation of Brusselator example. Places (object species) are depicted as circles, transitions (rules) are depicted as rectangles. The numbers on the arrows specify how many tokens (objects) are necessary in the relevant place before the transition becomes enabled. The inflow transition is shown in a lighter colour.

some inflow of the TM system). The initial configuration  $c_0$  is left unspecified for the moment. This system has the following stoichiometric matrix:

$$R_B = \begin{pmatrix} 1 & -1 & -2 & 3 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \quad (8)$$

This example will be further analyzed in the following sections. It is the equivalent of the Brusselator [25], a discrete model for the Belousov-Zhabotinskii chemical reaction, formulated as an open system (see Sect. 3.3). A Petri net representation (as a place-transition net) is given in Fig. 1.

Note that we can decompose the stoichiometric matrix uniquely as  $R = R^+ - R^-$ , with  $R^+, R^- \geq 0$  and  $\langle R_i^+, R_i^- \rangle = 0$  for all  $i$  columns. This decomposition is the main reason why we require the rules of  $\Pi$  to be nondegenerate.

From the definitions follows directly:

**Lemma 15** A cumulative vector  $k$ -trace  $v(\tau) = (v_0, \dots, v_k)$  is enabled for a configuration  $c_0$  iff  $c_0 + Rv_i \in \mathbb{N}_0^m$  for all  $0 \leq i \leq k$ .

A pseudo-trajectory  $t(c_0, \tau) = (c_0, \dots, c_k)$  of a TM system, given a  $k$ -trace  $\tau$  and an initial configuration  $c_0$ , is defined as follows:

$$c_{i+1} = c_i + r_{\tau_{i+1}} \quad (\text{single evolution step}) \quad (9)$$

which is equivalent to

$$c_i = c_0 + Rv_i \quad (\text{cumulative evolution step}) \quad (10)$$

where  $(v_0, \dots, v_k) = v(\tau)$ . It is a *trajectory* if  $\tau$  is enabled for  $c_0$ . The *application vector* of a (pseudo-) trajectory is  $|t(c_0, \tau)| = |\tau|$ .

We are interested in the following two problems:

**Problem 16** *Given a  $k$ -trace  $\tau = (\tau_1, \dots, \tau_k)$  in  $\Pi$ , characterize the set of all configurations of  $\Pi$  for which  $\tau$  is enabled.*

**Problem 17** *Given an application vector  $x \in \mathbb{N}_0^n$ , characterize efficiently the set of all configurations of  $\Pi$  for which there exists an enabled  $k$ -trace  $\tau$  with  $|\tau| = x$ .*

The first problem is easy, giving rise to the notion of *defect* (see Def. 35 below). The second problem is challenging: of course we could compute all possible permutations of the rules in the application vector and their defects (see Def. 36), but this would be too inefficient for practical use with large application vectors. An efficient algorithm for the second problem is presented in Section 3.8.

### 3.3 Open systems

In Example 14 we used a rule with reactant multiset empty, denoted by the symbol  $\lambda$  on the left-hand side of the rule. This was used to model some inflow. In this section we discuss the general use of in- and out-flows.

- Rules of the form  $r : \lambda \rightarrow p$ ,  $p \in \mathbb{N}_0^m$ , model *inflow* of some molecular species into the system, i.e. therewith it is possible to model some nutrient/energy supplies.
- Rules of the form  $r : q \rightarrow \lambda$ ,  $q \in \mathbb{N}_0^m$ , model *outflow* of some species of the system, i.e. the removal of some product or waste, or random degradation, i.e. a turnover effect.

We call a system with at least one in- or outflow rule an *open* system, since there is the possibility of interaction with the environment, in contrast to a *closed* system. Inflows (outflows) where the products (reactants) are of the form  $p = e_i$  ( $q = e_i$ ), for some unit vector  $e_i \in \mathbb{N}_0^m$ ,  $i \in \{1, \dots, m\}$ , are called *regular*.

Object species which partake both in a regular in- and outflow of the system, are called *buffered* (for details of buffering in continuous modelling see [30]).

With the following rules it is possible to reduce systems to a simpler form, retaining the essential dynamics. Topologically, i.e. with respect to the communicating classes, buffered species do not constrain the system. To be more precise, if an object species  $O$  is buffered, the number of objects  $O$  in the system can be increased or reduced (minimally to zero) by in- and outflow almost arbitrarily. Given a configuration  $c_0$ , the communicating class this belongs to is the same as for a configuration with a different number of objects  $O$ , but otherwise identical to  $c_0$ . The number of

objects of  $O$  therefore contains no information about the communicating class<sup>9</sup>, and we can remove this species from all rules, i.e. replace it with  $\lambda$  and eliminate all redundant rules created in the process.

**Criterion 18 (Rule of buffering)** *Buffered species can be removed from a system's definition.*

Furthermore, we can demand from the in- and outflow species that they interact immediately with the other species. Denote an object species  $i$  such that the  $i$ -th row of  $R$  contains only nonpositive (nonnegative) entries a *source* (*sink*).

**Criterion 19 (Rule of self-buffering)** *Inflow or outflow species which act only as sources or sinks, respectively, can be removed from a system's description.*

**Example 20** *Consider the system  $\Pi_B$  with  $m = 6$  and (invalid) rules*



*This is a discrete version of the Brusselator model, taken from [25]. It is usually assumed that this system has continuous inflow of species  $A$  and  $C$ , therefore we need the following additional rules  $r_5 : \lambda \rightarrow A$ ,  $r_6 : \lambda \rightarrow C$ . Furthermore we can assume continuous removal of the species  $E$  and  $F$  from the system, giving rise to additional rules  $r_7 : E \rightarrow \lambda$ ,  $r_8 : F \rightarrow \lambda$ . Rewriting the third rule according to our requirement of nondegeneracy (introducing a new object species  $G$ , making  $m = 7$ ) and reducing the system by applying Criterion 19 to species  $A$ ,  $E$  and  $F$ , we thus arrive at the following rule set:*



*Since  $C$  is buffered, we can apply Criterion 18, leaving us with*



*which is the system studied in Example 14, albeit with different names of the objects.*

---

<sup>9</sup> This is a symmetry argument: If two communicating configurations  $c, c' \in \mathbb{N}_0^m$  are identical up to one object, i.e.  $c_i = c'_i, \forall i \neq j$ , for some  $j \in \{1, \dots, m\}$ , it is uninformative to know the amount  $c_j$  of this object. Note that this argument does not hold if the object in question partakes in other rules, and its amount is close to the amount needed for some such rule. In practice, this case of limiting inflow is seldomly studied.

### 3.4 Compound space

Sometimes it is advantageous to rewrite a system  $\Pi$  to an “equivalent” system  $\Pi^C$  which simplifies the original dynamics, but increases the number of rules. We need the following definition:

**Definition 21** A compound is any non-empty multiset appearing on either the left- or right-hand side of a rule.

The set of all compounds is then  $C = \cup_{i \leq n} \{\max(0, r_i), -\min(0, r_i)\}$ . A compound is *trivial*, if it consists of one object species, with multiplicity one, only.

**Example 22** Continuing Example 14, we have the following compounds:  $c_1 = (1, 0, 0)^t$ ,  $c_2 = (0, 1, 0)^t$ ,  $c_3 = (0, 0, 1)^t$ ,  $c_4 = (2, 1, 0)^t$ ,  $c_5 = (3, 0, 0)^t$ , corresponding to  $A$ ,  $B$ ,  $C$ ,  $2A + B$  and  $3A$ , respectively. The compounds  $c_1$ ,  $c_2$  and  $c_3$  are trivial. Compound  $c_5$  is not trivial, as is compound  $c_4$ .

Note: The set of compounds is precisely the set of all columns of the matrices  $\{R^+, R^-\}$  in the unique decomposition of the given stoichiometric matrix into two non-negative matrices  $R^+, R^- \geq 0$ , where  $\langle R_i^+, R_i^- \rangle = 0$  for all  $i$ . (This correspondence is another reason why we require nondegenerate rules).

The idea to study compounds originates in CRNT [15], where the concept of *complex space* has been introduced, carrying the same linear structure as configuration space, albeit on a higher level of abstraction and with regards to rule applications. Since the name “complex space” is prone to confusion, we call this differently:

*Compound space* is the space generated by the compounds of the original system  $\Pi$  by *nonnegative integer linear combinations* (NILC, for short), i.e. the set of all sums  $\sum_{c \in C} n_c c$ , where  $n_c \in \mathbb{N}_0$ . In general, this space is smaller than configuration space.

**Proposition 23** Given a system  $\Pi$ , its set of compounds  $C = \{c_1, \dots, c_s\}$  generates all of  $\Pi$ 's configuration space iff it contains all unit vectors, i.e. iff all original objects occur as compounds.

**PROOF.** Sufficiency is trivial, since the unit vectors are part of configuration space. Necessity follows, since the set of all unit vectors generates all of configuration space.  $\square$

Since this condition is not always fulfilled, we have to include the original object species as trivial compounds. This is called *augmenting* the compounds and leads to the notion of *augmented space*, which is generated by nonnegative linear

combinations of the union of compounds and original object species. It is equal to configuration space by Proposition 23.

**Example 24** Consider a system  $\Pi$  of type  $(2, 1)$ . The single rule is given as  $r = (-2, 1)^t$ , so  $\Pi$ 's compound set is  $C = \{(2, 0)^t, (0, 1)^t\}$ , and does not contain  $(1, 0)^t$ . Therefore, a configuration of the form  $(2k_1 + 1, k_2)^t$ ,  $k_1, k_2 \in \mathbb{N}_0$  cannot be mapped to a configuration in compound space.

Given a system  $\Pi$ , we can consider its (augmented) *compound system*  $\Pi^C$ , which is itself a TM system, where the rules are given by (1) trivial dynamical rules only acting on the compounds, and (2) nontrivial conversion rules, transforming each compound into its constituent trivial compounds and vice versa.

**Example 25** Writing Example 14 as an (augmented) compound system  $\Pi^C = (m_C, \mathcal{R}_C, c_0)$ , we find  $m_C = 5$ , the following dynamical rules:  $r'_1 : \lambda \rightarrow c_1$ ,  $r'_2 : c_1 \rightarrow c_2$ ,  $r'_3 : c_4 \rightarrow c_3$ ,  $r'_4 : c_3 \rightarrow c_5$ , and the following conversion rules (and their inverses):  $r''_1 : c_4 \rightleftharpoons 2c_1 + c_2$ ,  $r''_2 : c_5 \rightleftharpoons 3c_1$ . Together they form the compound-stoichiometric matrix, denoted by  $R_C$ :

$$R_C = \left( \begin{array}{cccc|cccc} 1 & -1 & 0 & 0 & 2 & 3 & -2 & -3 \\ 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 \end{array} \right) \quad (11)$$

The horizontal line in the matrix separates the trivial compounds (first three rows) from the nontrivial compounds (last two rows). The vertical line separates the dynamical rules (first four columns) from the conversion rules (last four columns). As usual, the order of the rows and columns of  $R_C$  does not matter.

All equivalences between nontrivial compounds and their trivial constituent compounds are given by the communicating relation in  $\Pi^C$ .

### 3.5 Precycles and recurring configurations

**Definition 26** A pseudo-lattice digraph (PLD, for short) with local matrix  $R$  is a pair  $(R, D)$  such that:

- $R = (r_1 \cdots r_n)$  is a  $(m, n)$ -matrix with  $n$  columns  $r_1, \dots, r_n$ , having entries in  $\mathbb{Z}$ , for some  $m, n \in \mathbb{N}$ ,
- $D = (V, A)$  is a (finite or infinite) digraph, where
- $V \subseteq \mathbb{Z}^m$ , and

- $A \subseteq \{(u, u + r_i) \mid u \in V, 1 \leq i \leq n\}$ , such that each column  $r_i$  is used in at least one arc  $(u, u + r_i) \in A$  for some  $u \in V$ , and all  $i \leq n$ .

**Definition 27** A lattice digraph is a PLD  $(R, D)$ , with  $D = (V, A)$ , such that  $A = \{(u, u + r_i) \mid u \in V, 1 \leq i \leq n\}$ .

In particular, a lattice digraph is always infinite, *locally finite* and *n-regular*, i.e. the in- and out-degree of all vertices are equal to  $n$ .

The evolution digraph  $D$  is a pseudo-lattice digraph with the stoichiometric matrix as local matrix and vertices restricted to  $V \subseteq \mathbb{N}_0^n$ . In this section we analyze the relation between cycles in the evolution digraph and the following algebraic analog for pseudo-lattice digraphs:

**Definition 28** A precycle (in a PLD  $(R, D)$ ) is an application vector  $x \in \mathbb{N}_0^n$  such that  $Rx = 0$ .

The precycles are the elements of the nonnegative integer kernel  $\text{Ker}_{\mathbb{N}_0} R$  of the local matrix  $R$ . They form a submonoid  $\mathbf{PC}$  in  $\mathbb{N}_0^n$ : addition of two precycles is associative and leads to another precycle. Recall that the *inclusion* ordering of  $\mathbb{N}_0^n$  is given for two elements  $x, x' \in \mathbb{N}_0^n$  by  $x \leq x'$  iff  $x' - x \in \mathbb{N}_0^n$ . It is shown in [5] that each nonempty subset of  $\mathbb{N}_0^n$  has a finite subset of *minimal* elements in the inclusion ordering. Denote the set of minimal precycles by  $\mathbf{MPC}$ .

**Proposition 29** Each  $x \in \mathbf{PC}$  can be written as a (possibly nonunique) NILC of minimal precycles,

$$x = \sum_{y \in \mathbf{MPC}} a_y y, \quad a_y \in \mathbb{N}_0. \quad (12)$$

**PROOF.** Assume the contrary. The set  $S \subseteq \mathbf{PC}$  of all precycles which do not admit a representation as a NILC of elements of  $\mathbf{MPC}$  can be ordered by the inclusion ordering and has minimal elements. Let  $x \in S$  be such a minimal element. Obviously,  $x \in \mathbf{PC} \setminus \mathbf{MPC}$ . There then exists some  $y \in \mathbf{MPC}$  with  $x > y$ . Since  $Ry = Rx = 0$ , we have  $y_1 = x - y \in \mathbf{PC}$ , and  $y_1$  cannot be written as a NILC of elements of  $\mathbf{MPC}$  either. Since  $y_1 < x$ , this is a contradiction.  $\square$

The submonoid of precycles is therefore *finitely generated* by the minimal precycles via NILCs. The proof of Proposition 29 also shows how to find a representation as a NILC for a precycle  $x$  (in fact, all of them): Successively subtract minimal precycles  $y \in \mathbf{MPC}$  with  $y < x$  from  $x$ . This process always stops after a finite number of steps at the origin, such that the minimal precycles encountered along the way add up to a representation of  $x$ .

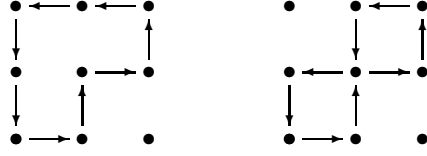


Figure 2. Two cycles in a pseudo-lattice digraph with the same application vector  $x = (2, 2, 2, 2)^t$ , where the local matrix has columns  $\{(1, 0)^t, (0, 1)^t, (-1, 0)^t, (0, -1)^t\}$ .

**Definition 30** Given a configuration  $c \in \mathbb{N}_0^m$ , and a nontrivial precycle  $x \in \mathbf{PC}$ , i.e.  $x \neq 0$ , the set of pseudo-cycles at  $c$  from  $x$  is  $\mathbf{PZ}(c, x) = \{t(c, \tau) \mid \tau \in \Xi(x)\}$ .

The set of pseudo-cycles at  $c$  is  $\mathbf{PZ}(c) = \bigcup_{x \in \mathbf{PC}} \mathbf{PZ}(c, x)$ . A pseudo-cycle at  $c$  is a cycle at  $c$ , if it is enabled. Denote the set of all cycles at  $c$  from  $x$  by  $\mathbf{Z}(c, x)$ . Denote by  $|z|$  the application vector of a (pseudo-) cycle  $z$  at  $c$ , i.e.  $|z| = |\tau|$  where  $z = t(c, \tau)$ . It is obvious that  $|z| \in \mathbf{PC}$  for all  $z \in \mathbf{Z}(c)$ . A (pseudo-) cycle  $z$  with  $|z| \in \mathbf{MPC}$  is called *minimal*. Denote the set of minimal cycles at  $c$  by  $\mathbf{MZ}(c)$ .

**Proposition 31** A minimal cycle is simple.

**PROOF.** Consider a cycle  $z = (c = c_0, c_1, \dots, c_k = c)$  of length  $k$  at some  $c \in \mathbb{N}_0^m$  from  $x \in \mathbf{MPC}$ . Then  $z = t(c, \tau)$ , for some  $\tau \in \Xi(x)$ , such that  $c_i = c_0 + Rv_i$  for  $i \in \{1, \dots, k\}$ , where  $(v_0, \dots, v_k) = v(\tau)$  is the cumulative vector trace of  $\tau$ . Assume  $z$  is not simple, then  $c_i = c_j$  for  $0 \leq i < j < k$ . Then  $z' = (c_i, \dots, c_j)$  is a cycle at  $c_i$  with application vector  $x' = v_j - v_i < v_k - v_0 = x$ , and  $x' \in \mathbf{PC}$ . This contradicts the minimality of  $x$ .  $\square$

**Example 32** Consider the pseudo-lattice digraphs with  $V \subseteq \mathbb{Z}^2$  and where the local matrix has columns  $\{(1, 0)^t, (0, 1)^t, (-1, 0)^t, (0, -1)^t\}$ . The minimal precycles are  $x_1 = (1, 0, 1, 0)^t$  and  $x_2 = (0, 1, 0, 1)^t$ , and obviously give rise to simple cycles. An example of such digraphs is shown in Figure 2. The cycle on the left is simple, but nonminimal, arising from the application vector  $2x_1 + 2x_2 = (2, 2, 2, 2)^t$ . On the right, a nonsimple, nonminimal cycle from the same application vector is shown.

The example shows that the converse of Proposition 31 is false, in general.

In the theory of dynamical systems, the notions of fixed and periodic points play an important role in the analysis of a system. The discrete analog are recurring configurations, i.e. cycles in the evolution digraph.

**Definition 33** A configuration  $c \in \mathbb{N}_0^m$  is recurring if there exists a cycle at  $c$ .

Finding the recurring configurations can be done in two steps: (i) find the non-negative, integer kernel  $\text{Ker}_{\mathbb{N}_0} R$  of the stoichiometric matrix  $R$ , and (ii) for each such precycle: characterize those configurations for which there exist enabled pseudocycles.

Note: If the rules were bidirectional, i.e. if we would consider the underlying graph<sup>10</sup> of the evolution digraph, the precycles would be the *integer* solutions  $z \in \mathbb{Z}^n$  of  $Rz = 0$ . One could then write each precycle *uniquely* as an integer linear combination of a finite set of generators [32, Ch. 4.1]. In the directional case we consider here, there are usually nontrivial relations among the elements of  $\mathbf{PC}$  which make a unique representation impossible, in general (see Example 39). Nevertheless, following Stanley [33, Ch. 4.6] it is possible to decompose the monoid  $\mathbf{PC}$  into *simplicial* submonoids with unique *quasi-generators*.

### 3.6 The incremental algorithm of Contejean and Devie

As seen in Section 3.5, finding the (minimal) precycles corresponds to the problem of solving the linear homogenous diophantine equation  $Rx = 0$  in terms of application vectors  $x \in \mathbb{N}_0^n$ . Background information and an effective algorithm for the determination of the minimal solutions of this equation in general can be found in [6], building on earlier work of Fortenbacher [5] and others.

A short sketch of the basic algorithm follows. Consider the pseudocode of Algorithm 1. The algorithm starts with the unit step vectors  $\{e_i \in \mathbb{N}^n \mid i \leq n\}$ , as starting elements, i.e. tentative solutions. In each iterative step it checks if the elements solve the equation  $Rx = 0$  (where in our case  $R$  will be the stoichiometric matrix of some TM system). Each of the solutions found is considered a minimal solution, since the algorithm starts at the origin and continuously increments all elements. Elements which dominate (in the inclusion ordering) any minimal element are removed. All remaining elements are then increased by one unit step, i.e. the application of one additional rule is considered, but only if they satisfy the generalized *Fortenbacher's restriction*. This is a geometric condition on the direction in which the additional rule moves the system from the current configuration. This change has to be in such a way that the system moves closer to the origin (measured by the projection on the current configuration vector, using the scalar product). Correctness and termination are proved in [6].

This basic algorithm can be further optimized by introducing a refined ordering on solutions, and Contejean and Devie finally arrive at an efficient stack-based implementation.

---

<sup>10</sup> The underlying graph of a digraph  $D = (V, A)$  is the graph  $G = (V, E)$  with vertex set  $V$  and edge set  $E = \{(u, v) \cup (v, u) \mid (u, v) \in A\}$ .

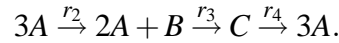
```

1: procedure CD( $R$ )
2:    $P \leftarrow \{e_1, \dots, e_n\}$             $\triangleright$  Start with unit step vectors, one for each rule.
3:    $B \leftarrow \emptyset$ 
4:   while  $P \neq \emptyset$  do
5:      $B \leftarrow B \cup \{x \in P \mid Rx = 0\}$             $\triangleright$  Found new minimal solution.
6:      $L \leftarrow \{x \in P \setminus B \mid \forall s \in B : x \not\geq s\}$             $\triangleright$  Guarantees minimality.
7:      $P \leftarrow \{x + e_i \mid x \in L, (Rx) \cdot (Re_i) < 0\}$             $\triangleright$  Fortenbacher's restriction.
8:   end while
9: end procedure

```

**Algorithm 1.** The incremental algorithm of Contejean & Devie. Input is a stoichiometric matrix  $R$ .

**Example 34** *Applied to Example 14, the above algorithm finds one minimal pre-cycle  $x_{min} = (0, 1, 1, 1)^t$ . This pre-cycle can occur, for example, as the following trajectory:*



### 3.7 Defects

We introduce the following simple, but crucial concept:

**Definition 35** *The defect of a  $k$ -trace  $\tau$  is*

$$\text{def}(\tau) = -\min\{0, Rv_1(\tau), Rv_2(\tau), \dots, R|\tau|\} \in \mathbb{N}_0^m. \quad (13)$$

The minimum in above definition is taken componentwise, of course, and the minus sign has been introduced such that the defect is nonnegative. The rationale behind this definition is as follows: Each component  $d_i$  of a defect  $d = \text{def}(\tau)$  contains the minimum number of objects of species  $i$  needed in a configuration to make the trace  $\tau$  possible, i.e. enabled. A natural way to compute the defect is to start at the origin of  $\mathbb{Z}^m$  and then apply each rule of the trace  $\tau$  under consideration, in order to find the minimal number of objects needed (a point in  $\mathbb{Z}^m$ ) which would make  $\tau$  enabled. This point is usually negative. See Figure 3 for the geometrical intuition.

**Definition 36** *Given a pre-cycle  $x \in \mathbb{N}_0^n$ , its defect is the set of minimal elements (in the inclusion ordering) in the set of defects of all multiset orderings  $\sigma \in \Xi(x)$  of  $x$ :*

$$\text{def}(x) = \text{min. elements} \{ \text{def}(\sigma) \mid \sigma \in \Xi(x) \} \subseteq \mathbb{N}_0^m \quad (14)$$

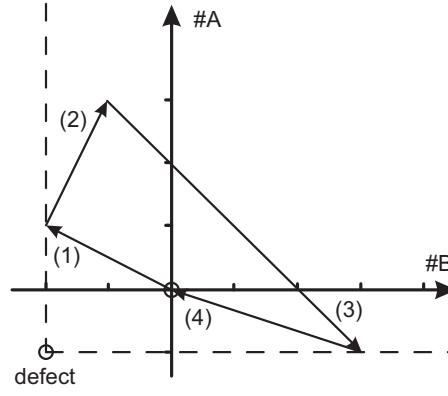


Figure 3. Example illustrating the meaning of *defect*. Configuration space is two-dimensional, and the pseudo-cycle shown results from the application of rules  $2B \rightarrow A$ ,  $\lambda \rightarrow 2A + B$ ,  $4A \rightarrow 4B$ ,  $3B \rightarrow A$ . The defect of this particular 4-trace is  $(1, 2)^t$  and its negation  $(-1, -2)^t$  has been marked in the figure.

**Example 37** The precycle  $x_{min}$  from Example 34 has a defect consisting of three elements:

$$\mathbf{def}(x_{min}) = \{(3, 0, 0)^t, (0, 0, 1)^t, (2, 1, 0)^t\}, \quad (15)$$

which results from, for example, the following 3-traces:  $\tau_1 = (2, 3, 4)$ ,  $\tau_2 = (4, 2, 3)$ , and  $\tau_3 = (3, 4, 2)$ .

**Definition 38** The defect space of a precycle  $x \in \mathbb{N}_0^n$  is the cone

$$\mathbf{ds}(x) = \mathbf{def}(x) + \mathbb{N}_0^m = \{v_1 + v_2 \mid v_1 \in \mathbf{def}(x), v_2 \in \mathbb{N}_0^m\}. \quad (16)$$

Defect space can also be characterized by

$$\mathbf{ds}(x) = \{c \in \mathbb{N}_0^m \mid c \geq y, \text{ for some } y \in \mathbf{def}(x)\} \quad (17)$$

Geometrically, it is the region of configuration space in which there exists some multiset ordering  $\sigma \in \Xi(x)$ , such that the system can cycle using the rules from  $x$  in the order given by  $\sigma$ . It is a finite union of *principal cones*, i.e. sets of the form  $\{c \in \mathbb{N}_0^m \mid c \geq c_0\}$ ,  $c_0 \in \mathbb{N}_0^m$ , with the minimal elements  $c_0 \in \mathbf{def}(x)$  as generators.

Consider an arbitrary precycle  $x \in \mathbb{N}_0^n$ . As seen in Proposition 29, this is a NILC

$$x = \sum_{y \in \mathbf{MPC}} a_y y_i, \quad a_y \in \mathbb{N}_0 \quad (18)$$

of minimal precycles. In general, such a representation is not unique.

**Example 39** Consider the system of type  $(2, 3)$ , where the rules are  $r_1 = (1, -1)^t$ ,  $r_2 = (2, -2)^t$ ,  $r_3 = (-4, 4)^t$ . Its minimal precycles are  $m_1 = (0, 2, 1)^t$ ,  $m_2 = (4, 0, 1)^t$ ,  $m_3 = (2, 1, 1)^t$ , given in some arbitrary order. We see that  $m_1 + m_2 = 2m_3$ , such that the precycle  $x = (4, 2, 2)^t$  has two different representations.

Uniqueness is guaranteed if the minimal precycles  $\mathbf{MPC}$ , considered as a set of vectors in  $\mathbb{R}^n$ , are linearly independent<sup>11</sup>. In this case we can define the *support*  $\mathbf{mpc}(x)$  of a precycle  $x \in \mathbf{PC}$ , which is the set of all minimal precycles in the above sum representation for  $x$  with positive coefficient. In the general case we can still talk about the *support*  $\mathbf{mpc}(x, a)$  of the representation of  $x$  in terms of  $a = (a_y)_{y \in \mathbf{MPC}}$ , where  $a$  is considered a function  $\mathbf{MPC} \rightarrow \mathbb{N}_0$ .

**Proposition 40** *Given an arbitrary precycle  $x \in \mathbb{N}_0^n$  and a representation  $x = \sum_{y \in \mathbf{MPC}} a_y y$  of it, we have*

$$\mathbf{ds}(x) \supseteq \bigcap_{y \in \mathbf{mpc}(x, a)} \mathbf{ds}(y) \quad (19)$$

**PROOF.** Consider two minimal precycles  $y_1, y_2 \in \mathbf{mpc}(x, a)$  which occur with constants  $a_1, a_2 \in \mathbb{N}$ . It is surely true that  $\mathbf{ds}(a_i y_i) \supseteq \mathbf{ds}(y_i)$  for  $i = 1, 2$ . Furthermore,  $\mathbf{ds}(y_1 + y_2) \supseteq \mathbf{ds}(y_1) \cap \mathbf{ds}(y_2)$ . With a standard theorem of set theory it follows that  $\mathbf{ds}(a_1 y_1 + a_2 y_2) \supseteq \mathbf{ds}(y_1) \cap \mathbf{ds}(y_2)$ . The proposition follows now by induction, the set  $\mathbf{mpc}(x, a)$  being finite.  $\square$

This result is obvious, since the intersection on the right hand side of Eq. 19 is the region of configuration space where all minimal precycles  $y \in \mathbf{mpc}(x, a)$  are enabled. Therefore  $x$  is also enabled, since one can iterate through the minimal precycles in any order. The question is, can one do better than that?

**Problem 41** *Given an arbitrary precycle  $x \in \mathbb{N}_0^n$ , and a representation  $x = \sum_{y \in \mathbf{MPC}} a_y y$  of it, under what conditions is it true that*

$$\mathbf{ds}(x) = \bigcap_{y \in \mathbf{mpc}(x, a)} \mathbf{ds}(y) ? \quad (20)$$

A different formulation of this problem is: When is  $\mathbf{ds}(y_1 + y_2) = \mathbf{ds}(y_1) \cap \mathbf{ds}(y_2)$  for all  $y_1, y_2 \in \mathbf{PC}$ ?

Note: If this holds for a given system, the defect of a general precycle would be just the union of all the (pair-by-pair) intersections (i.e. the coordinate-wise maxima) of the elements of the defects of its minimal precycles, since then

$$\mathbf{ds}(x_1 + x_2) = \{c \in \mathbb{N}_0^m \mid c \geq \max(y_1, y_2), \text{ for some } y_i \in \mathbf{def}(x_i), i = 1, 2\}, \quad (21)$$

<sup>11</sup> The precycles, being elements of the nonnegative integer kernel of  $R$ , measure the degree of dependency among the rules. Considered as a matrix  $C$  themselves, the nonnegative integer kernel of  $C$  measures the degree of dependency among the cycles, i.e. introduces a notion of “cycles of cycles”.

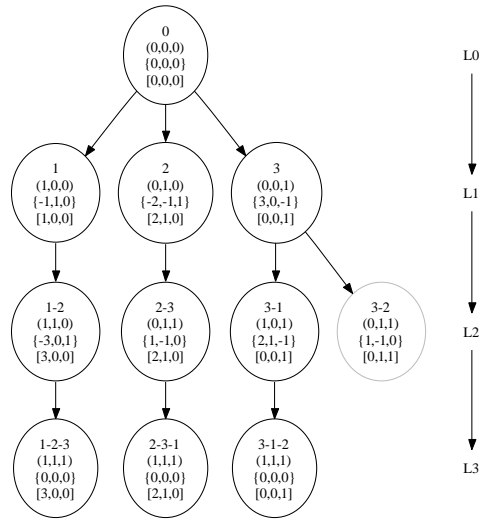


Figure 4. Defect tree of Brusselator example for the minimal precycle  $x = (1, 1, 1)^t$ . Nodes show, from top to bottom, (i) trace, (ii) application vector, (iii) configuration, (iv) defect. Truncated nodes, non-minimal in the tree order, have been shown in a lighter shade. Levels are indicated on the right.

such that

$$\mathbf{def}(x_1 + x_2) = \bigcup_{\substack{y_i \in \mathbf{def}(x_i), \\ i=1,2}} \max(y_1, y_2) \quad (22)$$

by de Morgan's laws. The commutativity and associativity of the maximum complete the inductive argument for the claim.

Since  $\mathbf{MPC} \subseteq \mathbf{PC}$ , the following result is also obvious:

**Lemma 42**

$$\bigcup_{x \in \mathbf{MPC}} \mathbf{ds}(x) \subseteq \bigcup_{x \in \mathbf{PC}} \mathbf{ds}(x) \quad (23)$$

### 3.8 Efficient calculation of defects

In this section we consider the problem posed in Section 3.2. The key to developing an efficient algorithm for the calculation of defects is the introduction of a suitable partial order on traces.

**Definition 43 (Tree order on traces)** Given two traces  $\tau_1, \tau_2$ , let

$$\tau_1 \leq \tau_2 \quad \text{iff} \quad (i) |\tau_1| = |\tau_2| \text{ and } (ii) \mathbf{def}(\tau_1) \leq \mathbf{def}(\tau_2). \quad (24)$$

The first condition guarantees that we are comparing traces with the same application vector only. Combined with a simple branch-and-bound scheme, Algorithm 2

```

1: procedure DEFECTS( $R, x$ )
2:    $\tau \leftarrow \emptyset$  ▷ Initial trace
3:    $c \leftarrow (0, \dots, 0)^t$  ▷ Initial configuration
4:    $d \leftarrow (0, \dots, 0)^t$  ▷ Initial defect
5:    $Q \leftarrow \mathbf{new}(node(\tau, c, d))$  ▷ Nodes contain traces, configuration, defect
6:   while  $Q \neq \emptyset$  do
7:      $P \leftarrow \{p \mid p \prec_x q, q \in Q\}$  ▷ One more rule application
8:     for all  $(p, p') \in P \times P$  and  $p \neq p'$  do
9:       if  $p \leq p'$  then ▷ Tree order
10:         $P \leftarrow P \setminus \{p'\}$  ▷ Remove non-minimal elements
11:       end if
12:     end for
13:      $Q \leftarrow \text{update}(P)$  ▷ Calculate new configuration and defect
14:   end while
15:   return  $P$  ▷ Set of minimal elements at last level
16: end procedure

```

**Algorithm 2.** The defect algorithm. Input is a stoichiometric matrix  $R$  and an application vector  $x$ .

then efficiently calculates defects for a known precycle  $x \in \mathbf{PC}$ . It is basically a hierarchical search, where each level corresponds to a distinct number of rule applications. Starting from the zero configuration, at each step the next level is generated by applying another rule to each configuration at the present level, for all (possible) rules. We use the symbol  $\prec_x$  to denote the *covering relation* on traces with respect to an application vector  $x$ , i.e.  $p \prec_x q$  only if  $p$  arises from  $q$  by applying exactly one more rule from the application vector, such that  $|p| \leq |x|$ . The configurations so generated are then compared under the tree order and non-minimal branches of the tree are cut off. If two configurations are equal under the tree order, one of them is also (randomly) cut off. The remaining configurations at the last level are the searched for defects.

**Example 44** For the Brusselator system from Example 14 and its minimal precycle  $x = (1, 1, 1)^t$  the defect tree is shown in Fig. 4. The precycle  $2x = (2, 2, 2)^t$  has the defect tree shown in Fig. 5. As can be seen, the defect of both precycles is the same, so in this case Equation 20 holds.

### 3.9 Conservation of mass

Another application of Algorithm 1 is to check a TM system for mass conservation. Chemical systems are usually subject to the constraint of conserving mass in their reactions. Formally this means that there exists a *mass function*  $z : \{1, \dots, m\} \rightarrow \mathbb{R}_+ \setminus \{0\}$  which assigns a fixed, positive mass to each object species. For convenience we identify this function with a *mass vector*  $z = (z_1, \dots, z_m)^t \in \mathbb{R}_+^m \setminus \{0\}$ .

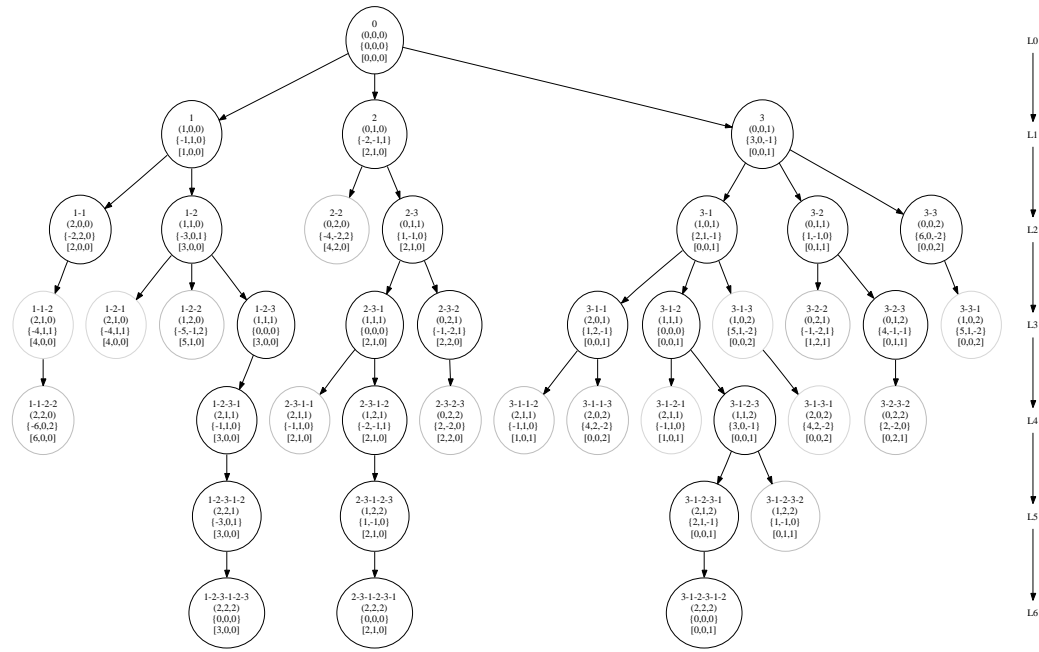


Figure 5. Defect tree of Brusselator example for the precycle  $x' = (2, 2, 2)^t$ . Nodes show, from top to bottom, (i) trace, (ii) application vector, (iii) configuration, (iv) defect. Truncated nodes, non-minimal or non-unique in the tree order, have been shown in lighter shades. Levels are indicated on the right.

**Definition 45** The mass of a configuration  $c \in \mathbb{N}_0^m$  is  $m(c) = \langle c, z \rangle$ .

Conservation of mass is enforced by requiring  $\langle q, z \rangle = \langle p, z \rangle$  for each rule  $r : q \rightarrow p$ . Since rules are nondegenerate, this can be written in a more compact way:

**Definition 46** A stoichiometric matrix  $R$  is mass-conserving if

$$z^t R = 0 \quad (25)$$

for some nontrivial mass vector  $z \in \mathbb{R}_+^m$ ,  $z > 0$ , where  $z^t$  denotes the transpose of the vector  $z$ .

By transposing the whole expression, we can also write this condition as  $R^t z = 0$ . Now it is easy to see if a given TM system can be construed as a realistic model for biochemical reactions, since above equation has at least one nontrivial solution iff  $R^t$  has a nontrivial (real) kernel.

Note that apart from transposition and working in the reals, instead of the integers, this is almost the same condition as in the characterization of the precycles. Therefore, if we consider the special situation where we restrict the mass vector to be rational, i.e. a positive, integer multiple of some smallest mass  $z_0 \in \mathbb{R}_+ \setminus \{0\}$ , we can use Algorithm 1 to find all solutions.

**Proposition 47** If two states  $c, c'$  in a mass-conserving system are communicating,

they have the same mass:  $c \sim c' \Rightarrow m(c) = m(c')$ .

**PROOF.** If  $c \sim c'$  there exists a path  $x$  from  $c$  to  $c'$ . Its application vector  $|x| \in \mathbb{N}_0^n$  fulfills  $c' = c + R|x|$ , such that  $m(c') - m(c) = (R|x|)^t z = |x|^t R^t z = 0$ .

## 4 Communicating classes

In Section 4.1 the free regime is introduced, where all rules are always enabled. Criteria are given which allow to check when there are transients or a global communicating class. In Section 4.2 we show that testing for communication in the free regime can be done efficiently.

### 4.1 The free regime

In this section we want to consider different questions. Call  $(\mathbb{Z}^m, +)$  *free space* and consider the behaviour of a TM system  $\Pi$  in this space, if all rules are always enabled a priori, i.e. the system can “borrow” objects as much as it wants. This corresponds to considering  $\Pi$  in the *free regime*, where configurations are far from the origin, such that all rules are automatically enabled. Defects do not play any role in this regime, and pseudo-cycles are always cycles. This is the situation referred to at the end of Section 3.5. The notions from Section 2.4, formulated for  $(\mathbb{N}_0^m, +)$  originally, carry over to the space  $(\mathbb{Z}^m, +)$  easily.

The evolution digraph in free space is, by definition, a lattice digraph. It can also be characterized as follows: Define a *translation* by  $x \in \mathbb{Z}^m$  of a subset  $S \subseteq \mathbb{Z}^m$  by  $S + x = \{s + x \mid s \in S\}$ . Define the translation  $\tau_x(R, D) = (R, \tau_x D)$  by  $x \in \mathbb{Z}^m$  of a pseudo-lattice digraph  $(R, D)$  with  $D = (V, A)$  by  $\tau_x D = (\tau_x V, \tau_x A)$ , where  $\tau_x A = \{(u + x, v + x) \mid (u, v) \in A\}$ . A pseudo-lattice digraph  $(R, D)$  with  $D = (V, A)$  is called *isotropic* if it is invariant to all translations along its arcs, where a *translation along an arc*  $(u, v) \in A$  is a translation by  $v - u \in \mathbb{Z}^m$ . Obviously, a pseudo-lattice digraph is a lattice digraph iff it is isotropic (since all columns from the local matrix are used in some arc, by definition).

**Problem 48** *How many communicating classes exist in the free regime?*

We have to distinguish two cases. There might be rules which do not occur in any cycle of the system. Therefore, these rules lead the system irreversibly from one communicating class to another. Furthermore, they can be repeatedly applied (since the free lattice<sup>12</sup> of configurations is translation-invariant) and thus there exists an

<sup>12</sup> We use the word *lattice* here in its geometrical meaning, not in the order-theoretic sense.

infinite number of asymptotically different communicating classes. Looking at the reduced (free) configuration space, it will be infinite.

**Definition 49** *A rule is transient, if it is not part of any precycle.*

Otherwise, a rule is called *reversible*, since by following the rest of any (permutation of a) cycle in which the rule under consideration partakes, we can reach the origin again. A reversible rule thus has at least one (formal) inverse.

Since cycles are equivalent to pseudo-cycles in the free regime, and all pseudo-cycles are generated by the minimal precycles via their multiset orderings, we only need to check this condition on the minimal precycles.

**Definition 50** *A matrix  $R$  is reversible, if its columns, considered as rules, are reversible. Otherwise, it is transient.*

**Proposition 51** *If the local matrix of a lattice digraph  $D$  is reversible, the communicating classes of  $D$  are independent<sup>13</sup>.*

A reversible set of rules guarantees that the communicating classes in the free evolution digraph are independent. It also means that we now have a group structure for the rules, since each rule has an inverse; thus reachability is equivalent to communication. But there is still the second possibility, namely, that more than one independent communicating class exists. This would mean that there are disconnected dynamical regimes, which can usually be considered unrealistic.

**Definition 52** *The reduced free configuration space of a TM system  $\Pi$  is the quotient  $(\mathbb{Z}^m)_{\sim} = \mathbb{Z}^m / \sim / \sim_t$ .*

**Problem 53** *Under which conditions is the reduced free configuration space of a TM system  $\Pi$  finite?*

This problem is very hard in general, since it corresponds to a well known problem in multidimensional crystallography, which has seen no general, effective solution so far.

The following special case is the most important one for our applications<sup>14</sup>:

**Proposition 54** *Reduced free configuration space  $(\mathbb{Z}^m)_{\sim}$  is trivial, i.e., consists of only one communicating class, iff the origin is equivalent under the communication*

---

Consult [35, Ch. 7] for more on this geometric aspect of Molecular Dynamics.

<sup>13</sup> The existence of two or more independent communicating classes means that the corresponding topological Markov chain  $F_{ij}$  has a block structure. Furthermore, this means that there will not be a unique stationary distribution [3].

<sup>14</sup> The existence of exactly one communicating class means that the corresponding topological Markov chain  $F_{ij}$  is irreducible, i.e. there exists a positive integer  $k$  such that  $F_{ij}^k > 0$ .

relation to all unit vectors and their negative counterparts:

$$0 \sim E_+ \cup E_-, \quad (26)$$

where  $E_{\pm} = \{\pm e_i \mid 1 \leq i \leq m\}$ .

**PROOF.** (Similar to the proof of Prop. 23.)

Sufficiency is shown first: If the unit vectors and their negative copies are equivalent to the origin, this means they are reachable by cycles in free space. Since free space is isotropic, we can move the origin to one of these vectors. Iterating this argument shows that we can reach all of free space, therefore it becomes trivial under the reachability equivalence relation.

Necessity is immediate, since the unit vectors and their counterparts trivially are elements of free space.  $\square$

This gives an effective and efficient test for triviality by solving the  $2m$  inhomogeneous diophantine equations  $Rx = \pm e_i$ ,  $i \leq m$ , which can be achieved by a modification of Contejean and Devie's original algorithm, as also described in [6]:

Introduce an extra variable  $x_0 \in \mathbb{N}_0$  with  $r_0 = \mp e_i$  as new first column of the otherwise identical stoichiometric matrix  $R$  (note the change of sign). In Algorithm 1, whenever the coordinate associated with  $x_0$  reaches the value 1, it is *frozen*, i.e. not allowed to be incremented any further.

An application of this criterion can be seen in the example in Section 5.2.

#### 4.2 Testing for communication

The aim of this section is to show that one can efficiently test for communication. The test relies on the reversibility of rules, i.e. it is only applicable if the stoichiometric matrix is reversible. This is no restriction, though, since by using transient rules one can never realize communication, only reachability. Thus, if the stoichiometric matrix is transient, one can remove the transient rules from it and then apply the test described below. We therefore assume a reversible matrix throughout the following.

If the stoichiometric matrix  $R$  is invertible (over  $\mathbb{R}$ ), we can use its inverse  $R^{-1}$  to check for communication: Two states  $c, c' \in \mathbb{Z}^m$  are communicating iff  $R^{-1}(c' - c)$  is *integral*, i.e. iff  $R^{-1}(c' - c) \in \mathbb{Z}^m$ . Of course, it is usually not the case that  $R$  is invertible. To proceed, one would like to (i) remove linearly dependent rules, (over  $\mathbb{Z}$ ), and (ii) add enough, possibly none, linearly independent rules  $r'_1, \dots, r'_k$ , arriving at an invertible matrix  $R'$  (after the example given below we explain why this is

usually not possible, though). Two states  $c, c' \in \mathbb{Z}^m$  are then communicating iff  $(R')^{-1}(c' - c) \in \mathbb{Z}^m$ , and such that the components of  $(R')^{-1}(c' - c)$  corresponding to the added rules are zero. The latter requirements are called *conservation laws*.

**Example 55** Consider the stoichiometric matrix of Example 14, with the first rule removed (for the purpose of this example), i.e. with no inflow. Since we still have a precycle  $x = (1, 1, 1)^t$ , the rules are not linearly independent over  $\mathbb{Z}$ . Removing the last rule (or any other, for that matter), the rank is only two, so we add an additional rule  $r' = (0, 0, 1)^t$ . Now

$$R = \begin{pmatrix} -1 & -2 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad \text{with inverse} \quad R^{-1} = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} & 0 \\ -\frac{1}{3} & -\frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 1 \end{pmatrix}. \quad (27)$$

A configuration  $c \in \mathbb{Z}^3$  is communicating with the origin iff  $R^{-1}c = (x_1, x_2, x_3)^t$  is integral, and such that  $x_3 = 0$ . The latter means that  $c_1 + c_2 + 3c_3 = 0$  is a conservation law. For example,  $c = (3, 0, -1)^t$  is communicating with the origin, since  $R^{-1}c = (-1, -1, 0)^t$ . On the other hand,  $c' = (4, 0, -1)^t$  lies in a different communicating class, since  $c'$  does not fulfill above conservation law:  $c'_1 + c'_2 + 3c'_3 = 1 \neq 0$ .

The problem with this method is the removal of the linearly dependent rules. A set of rules  $\{r_1, \dots, r_k\}$  is *linearly dependent over*  $X \subseteq \mathbb{R}$  if  $\lambda_1 r_1 + \dots + \lambda_k r_k = 0$  has at least one nontrivial solution  $\lambda_i \in X$ ,  $1 \leq i \leq k$ , where not all  $\lambda_i$  are zero. Since  $\mathbb{Z} \subseteq \mathbb{R}$ , linear dependence over  $\mathbb{Z}$  implies linear dependence over  $\mathbb{R}$ , but the opposite is false in general. In particular, we can have linearly dependent rules over  $\mathbb{R}$ , that are independent over  $\mathbb{Z}$ . In this case, the method fails. Moreover, testing for linear independence over  $\mathbb{Z}$  is difficult.

Both problems are overcome by considering the *Hermite normal form* of the matrix  $R$ . This is the unique matrix  $(B \ 0)$  arising from  $R$  by elementary integer column operations, where  $B$  is a nonsingular, nonnegative matrix in which each row has a unique maximum entry, which is located on the main diagonal of  $B$ . The Hermite normal form of a matrix  $R$  exists iff  $R$  has full row rank (over  $\mathbb{R}$ ).

**Proposition 56** Two configurations  $c, c' \in \mathbb{Z}^m$  are communicating iff

$$B^{-1}(c' - c) \in \mathbb{Z}^m. \quad (28)$$

**PROOF.** Since  $B$  is nonsingular, its inverse  $B^{-1}$  exists. Furthermore,  $b = c' - c$  belongs to  $\text{im}_{\mathbb{Z}} R = \{Rx \mid x \in \mathbb{Z}^n\}$  iff  $B^{-1}b \in \mathbb{Z}^m$  [32, Ch. 4.1].

**Example 57** Continuing Example 14, we replace the first column of  $R$ , the inflow

rule  $r_1 = (1,0,0)^t$ , by  $r' = (2,0,0)^t$  (for the purpose of this example). The rule  $r'$  is not reversible, so we also assume a corresponding outflow  $r'' = (-2,0,0)^t$ . The Hermite normal form of the new stoichiometric matrix  $R' = (r' r'' r_2 r_3 r_4)$  is  $(B' 0)$ , where

$$B' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 2 \end{pmatrix}, \quad \text{with inverse} \quad (B')^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix}. \quad (29)$$

The configuration  $c_1 = (3,0,-1)^t$  is communicating with the origin according to Proposition 56, since  $(B')^{-1}c_1 = (3,0,-2)^t$  is integral. The configuration  $c_2 = (4,0,-1)^t$ , on the other hand, lies in a different communicating class, since  $(B')^{-1}c_2 = (4,0,-\frac{5}{2})^t$  is not integral.

Since the computation of the Hermite normal form can be done in polynomial time [32, Ch. 5.3], we have an efficient test for communication.

**Proposition 58** *The communication relation between two states  $c, c' \in \mathbb{Z}^m$  can be evaluated in polynomial time.*

From Proposition 56 the following is obvious:

**Proposition 59** *Reduced free configuration space  $(\mathbb{Z}^m)_\sim$  is trivial, iff the stoichiometric matrix  $R$  has Hermite normal form  $(B 0)$ , where  $B$  is the identity matrix of  $\mathbb{Z}^m$ .*

**Example 60** *Considering the open Brusselator system from Example 14, we see that  $R$  is a  $(3,4)$ -matrix with full row rank. It is not reversible, though, so for the purpose of this example we assume that there is a corresponding outflow  $r'_1 = (-1,0,0)^t$  for the inflow  $r_1 = (1,0,0)^t$ . The matrix  $R' = (r'_1 r_1 r_2 r_3 r_4)$  has Hermite normal form<sup>15</sup>  $(B 0)$ , where  $B = \text{diag}(1, 1, 1)$  is the identity matrix in  $\mathbb{Z}^3$ . This shows that all of free space is reachable from each configuration  $c \in \mathbb{Z}^m$ , and the existence of one global communicating class is established.*

If the stoichiometric matrix does not have full row rank, one can complete the stoichiometric matrix to a generating matrix<sup>16</sup>  $R'$  of  $\mathbb{R}^m$ . This can be done easily, since the calculation of the rank can be done efficiently (for example by Gaussian elimination) and by the replacement theorem of Steinitz [31] one needs to consider the unit vectors from  $E_+$  (or any other basis of  $\mathbb{R}^m$ ) as candidates only.

<sup>15</sup> The Hermite normal form in the example has been computed with the computer algebra system Maple<sup>TM</sup>.

<sup>16</sup> A  $(m,n)$ -matrix  $R$  is called a generating matrix of  $\mathbb{R}^m$  if it has full row rank, or equivalently if  $\text{im}_{\mathbb{R}} R = \{Rx \mid x \in \mathbb{R}^n\} = \mathbb{R}^m$ . It will be a basis of  $\mathbb{R}^m$  if all the columns are linearly independent (over  $\mathbb{R}$ ), i.e. if  $n = m$ .

$A + B \xrightarrow{r_1} C$	$C \xrightarrow{r_2} A + B$
$C \xrightarrow{r_3} D$	$D \xrightarrow{r_4} A + E + F$
$F \xrightarrow{r_5} G$	$G \xrightarrow{r_6} H + I$
$I \xrightarrow{r_7} B$	
$E + J \xrightarrow{r_8} K$	$K \xrightarrow{r_9} E + J$
$H + J \xrightarrow{r_{10}} L$	$L \xrightarrow{r_{11}} H + J$
$K \xrightarrow{r_{12}} E + M$	$M \xrightarrow{r_{13}} O$
$L \xrightarrow{r_{14}} H + N$	$N \xrightarrow{r_{15}} P$
$E \xrightarrow{r_{16}} [T]$	$O \xrightarrow{r_{17}} [R]$
$H \xrightarrow{r_{18}} [X]$	$P \xrightarrow{r_{19}} [S]$
$O + U \xrightarrow{r_{20}} V$	$V \xrightarrow{r_{21}} O + [W]$
$P \xrightarrow{r_{22}} U + Q$	$Q \xrightarrow{r_{23}} P$

Table 1  
Symbolic rules of the LacZ/LacY system.

Object name	B	J	O	P	T	X
Biological entity	RNAP	Ribosome	LacZ	LacY	dgrRbsLacZ	dgrRbsLacY
	Q	W	E	H	R	S
	lactose product		RbsLacZ	RbsLacY	dgrLacZ	dgrLacY

Table 2  
Names of corresponding biological entities, i.e. molecules and protein complexes (Ribosome).

An example of such a completion is given in Section 5.2.

## 5 Biological example: LacZ-LacY regulation

### 5.1 Precycles of the basic model

We consider the following model for the expressivity of LacZ/LacY proteins in *E. coli* bacteria, taken from [34]. Its rules are shown in Table 1. Biological species names have been replaced by capital letters. The correspondence for the most important reactants has been given in Table 2.

More information on the biological meaning of these objects and some stochastic simulation results can be found in [34]. In the equations we have already indicated by bracketing the outflows and introduced an extra variable  $Q$  to ensure nondegen-

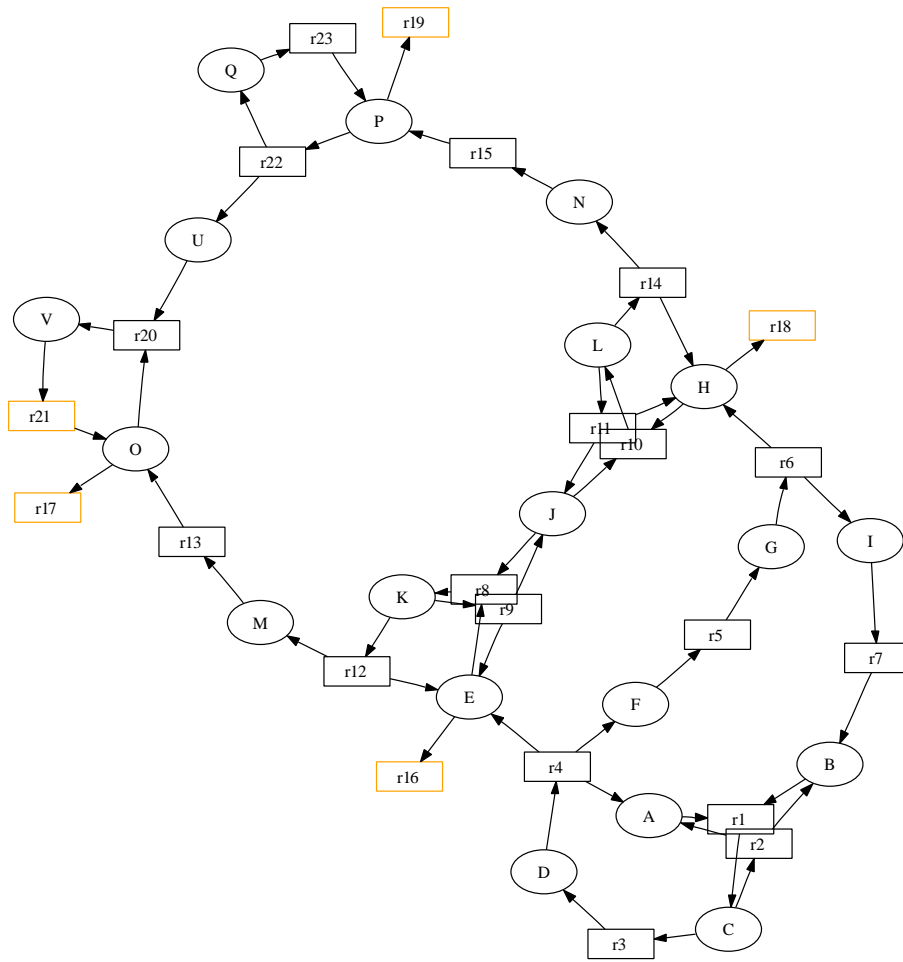


Figure 6. Petri net representation of LacZ/LacY example. Outflow species (see text) are shown without places and in a lighter colour.

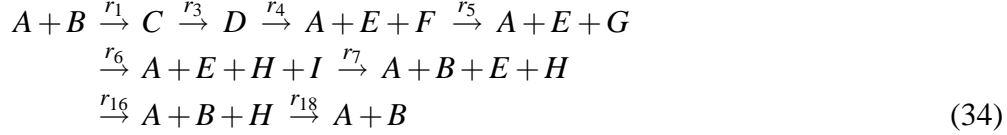
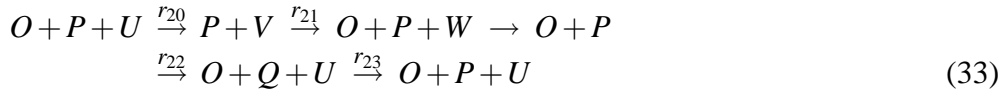
eracy. In our version of the model the outflows will be replaced by rules of the form  $O \rightarrow \lambda, P \rightarrow \lambda$ , etc. Figure 6 shows a Petri net representation of the system, in which the complicated dependencies of the objects are depicted. Outflow species have been shown without places to make it more readable.

Without outflows, the system will have the following three internal cycles, given in some arbitrary trace sequence:



Applying Algorithm 1 to the open system gives us the following additional possi-

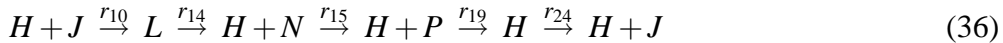
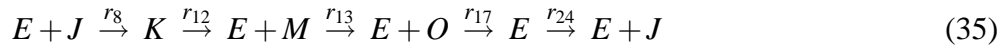
bilities<sup>17</sup> :



The first one is the primal *production* cycle of the system, with object species  $W$  representing the final product, species  $O$  and  $P$  being the LacZ and LacY proteins, respectively. The second one is the ribosomal *degradation* cycle, with species  $E, H$  representing degraded LacZ and LacY.

## 5.2 Communicating classes in the LacZ-LacY system

We will check for triviality of the set of communicating classes in above model. Since not all rules are part of some cycle, this is an indication that the model is not properly posed. In particular, there might be some inflows missing. We therefore, with some biological hindsight, include an inflow for species  $J$ , representing Ribosome entities that can be used up. With the additional rule  $r_{24} : \lambda \rightarrow J$  we find the following additional solutions:



These are primary degradation cycles for the LacZ and LacY proteins, where proteins are disabled by turnover (i.e. they degrade spontaneously).

Now the system is reversible and we can test for a global communicating class. Furthermore, we can concentrate on the internal behaviour:

**Criterion 61 (Rule of biological recycling)** *Assume (for biological systems) that each inflow and outflow species is part of some further reaction cycle (in a larger system).*

<sup>17</sup> Note: Most reaction cycles in biology are rather small, given the constant threat of turnover, i.e. spontaneous degradation of biologically active molecules.

In the triviality check according to Prop. 54 we therefore only need to check the existence of a solution  $x \in \mathbb{N}_0^m$ ,  $Rx = \pm e_i$ , for all  $i \leq m$  which do not correspond to one such inflow/outflow object.

Applying this criterion and running the modified algorithm gives no solution  $x \in \mathbb{N}_0^m$  for  $Rx = e_{\{A\}}$ . In fact, the following positive unit vectors are all not reachable:

$$e_{\{A\}}, e_{\{B\}}, e_{\{C\}}, e_{\{D\}}, e_{\{F\}}, e_{\{G\}} \text{ and } e_{\{I\}}.$$

Due to the symmetry between positive and negative unit vectors in free space, their negative counterparts are also not reachable. The system thus has an infinite number of disconnected communicating classes. The stoichiometric matrix has rank 17, so there are seven linearly dependent rules (over  $\mathbb{R}$ ) from the 24 in total.

**Definition 62** A set  $S = \{r_{i_1}, \dots, r_{i_k}\}$  of columns from the stoichiometric matrix  $R$  is an independent blocking set of  $R$ , if each column  $r_{i_j} \in S$  occurs in exactly one minimal precycle:  $(x)_{i_j} > 0$  for exactly one  $x \in \mathbf{MPC}$ , for all  $j \leq k$ .

An independent blocking set thus is a set of rules  $S$ , one from each precycle, and linear dependent (over  $\mathbb{Z}$ ) with  $R \setminus S$ , such that removing a rule  $r \in S$  from  $R$  does not destroy the linear dependence of the remaining set  $S \setminus \{r\}$  with  $R \setminus S$ . In other words, if  $S$  is an independent blocking set of  $R$ , then  $S \setminus \{r\}$  is an independent blocking set of  $R \setminus \{r\}$ , for all  $r \in S$ . Note that for a general system one usually does not expect to have an independent blocking set.

There exists an independent blocking set for the LacZ-LacY model, for example  $S = \{r_2, r_4, r_9, r_{11}, r_{12}, r_{19}, r_{22}\}$ . These seven rules can be removed from the stoichiometric matrix, which then has rank 17. To make the matrix nonsingular, we add  $e_1 = (1, 0, \dots, 0) \in \mathbb{Z}^{19}$  and  $e_2 = (0, 1, 0, \dots, 0) \in \mathbb{Z}^{19}$  as columns. Its inverse is then easily computed (not given here) and has only integer entries. The rows corresponding to the rules  $e_1$  and  $e_2$  give us two independent conservation laws (where  $c_A$  stands for the number of objects from species  $A$  in a configuration  $c \in \mathbb{Z}^{19}$  under consideration, etc.):

$$c_A + c_C + c_D = 0 \tag{37}$$

$$c_B + c_C + c_D + c_F + c_G + c_I = 0 \tag{38}$$

We see that in the free regime each configuration  $c \in \mathbb{Z}^{19}$  is reachable from the origin, iff it fulfills these two equations.

Of course, these results are just the starting point for further, more elaborate analysis of this model and its dynamical behaviour.

## 6 Discussion

We have seen that cycles in chemical reaction networks and membrane systems can be effectively calculated with standard algorithms. This information is useful for model checking, as can be seen in the biological example, where we found the well-known internal as well as production/degradation cycles and had to expand the model appropriately to find further important cycles which were missing in the original formulation. It can also be checked whether or not there will be essentially disconnected communicating classes, by looking for transient rules and then solving for positive and negative unit vectors. This information is crucial for a full probabilistic treatment in the context of Markov chain theory. The notions of precycles and their defects, which have been introduced, lead to some interesting mathematical structures and some important open problems.

We hope that the membrane systems community finds this work stimulating for their own research and will make use of the concepts introduced here<sup>18</sup>.

For completeness we should also mention the related work of [14] which is concerned with finding similar cycle bases for chemical reaction networks, and the theory of ear-decompositions of directed graphs [17].

## Acknowledgement

M. Muskulus and R. Brijder acknowledge support by the Nederlandse Wetenschappelijke Organisatie NWO under grant no. 635.100.006. The work of D. Besozzi and D. Pescini has been supported by the European Research Training Network “Segravis”. The authors acknowledge the use of the graph visualization software Graphviz [10].

## References

- [1] J. Bang-Jensen, G. Gutin, *Digraphs. Theory, Algorithms and Applications*, Springer, 2002.
- [2] L. Bianco, F. Fontana, V. Manca, P systems with reaction maps, *Int. J. Found. Comp. Sci.* 17 (1) (2006) 3–26.

---

<sup>18</sup> The software used in the calculation of precycles and defects will be available on the author’s webpage (<http://www.math.leidenuniv.nl/~muskulus/>).

- [3] P. Brémaud, Markov Chains. Gibbs Fields, Monte Carlo Simulation, and Queues, Vol. 31 of Texts in Applied Mathematics, Springer, 1999.
- [4] P. Buchholz, P. Kemper, Hierarchical reachability graph generation for Petri nets, *Formal Methods in System Design* 21 (2002) 281–315.
- [5] M. Clausen, A. Fortenbacher, Efficient solution of linear diophantine equations, *J. Symbolic Computation* 8 (1989) 201–216.
- [6] E. Contejean, H. Devie, An efficient incremental algorithm for solving systems of linear diophantine equations, *Information and Computation* 113 (1994) 143–172.
- [7] B. A. Davey, H. A. Priestley, *Lattices and Order*, 2nd Edition, Cambridge University Press, 2002.
- [8] R. Diestel, *Graph Theory*, 3rd Edition, Springer, 2005.
- [9] P. Dittrich, P. S. di Fenizio, Chemical organization theory: towards a theory of constructive dynamical systems, Tech. rep., arXiv preprint: <http://lanl.arxiv.org/abs/q-bio.MN/0501016> (2005).
- [10] E. R. Gansner, S. C. North, An open graph visualization system and its applications to software engineering, *Software-Practice and Experience* 30 (11) (2000) 1203–1233.
- [11] M. A. Gibson, J. Bruck, Efficient exact stochastic simulation of chemical systems with many species and many channels, *J. Phys. Chem. A* 104 (2000) 1876–1889.
- [12] D. T. Gillespie, Exact stochastic simulation of coupled chemical reactions, *J. Phys. Chem.* 81 (1977) 2340–2361.
- [13] D. T. Gillespie, Approximate accelerated stochastic simulation of chemically reacting systems, *J. Chem. Phys.* 115 (2001) 1716–1733.
- [14] P. M. Gleiss, P. F. Stadler, A. Wagner, D. A. Fell, Relevant cycles in chemical reaction networks, *Adv. Complex Systems* 4 (2001) 207–226.
- [15] J. Gunawardena, Chemical reaction network theory for in-silico biologists, Tech. rep., Bauer Center for Genomics Research, Harvard University (June 2003).
- [16] A. Leporati, G. Mauri, C. Zandron, Quantum sequential P systems with unit rules and energy assigned to membranes, in: R. Freund, G. Păun, G. Rozenberg, A. Salomaa (Eds.), *Membrane Computing*, 6th International Workshop, WMC 2005, Vol. 3850 of LNCS, Springer, 2006, pp. 310–325.
- [17] M. Loeb, M. Matamala, Some remarks on cycles in graphs and digraphs, *Discrete Mathematics* 233 (2001) 175–182.
- [18] E. W. Mayr, An algorithm for the general Petri net reachability problem, in: *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, ACM Press, 1981, pp. 238–246.
- [19] A. Miner, D. Parker, Symbolic representation and analysis of large probabilistic systems, in: C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, M. Siegle (Eds.), *Validation of Stochastic Systems*, Vol. 2925 of LNCS, 2004, pp. 296–338.

- [20] T. Murata, Petri nets: Properties, analysis and applications, Proc. Inst. Electr. Eng. 77 (1989) 541–580.
- [21] M. Muskulus, R. Brijder, First steps toward a geometry of computation, in: M. A. Gutiérrez-Naranjo, A. Riscos-Núñez, F. J. Romero-Campero (Eds.), Third Brainstorming Week on Membrane Computing, Fenix Editora, Sevilla, 2005, pp. 197–218.
- [22] M. Muskulus, R. Brijder, Complexity of biocomputation: Symbolic dynamics in membrane systems, Int. J. Found. Comp. Sci. 17 (1) (2006) 147–165.
- [23] G. Păun, Computing with membranes, J. Comput. Syst. Sci. 61 (2000) 108–143.
- [24] G. Păun, Membrane Computing: An Introduction, Natural Computing, Springer, 2002.
- [25] D. Pescini, D. Besozzi, G. Mauri, C. Zandron, Analysis and simulation of dynamics in probabilistic P systems, in: A. Carbone, N. Pierce (Eds.), DNA Computing, 11th International Workshop on DNA Computing, LNCS 3892, Springer, 2005, pp. 236–247.
- [26] D. Pescini, D. Besozzi, G. Mauri, C. Zandron, Dynamical probabilistic P systems, Int. J. Found. Comp. Sci. 17 (2006) 183–204.
- [27] J. L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, 1981.
- [28] C. Reutenauer, Aspects mathématiques des Réseaux de Pétri, Masson, 1989.
- [29] C. H. Schilling, B. O. Palsson, The underlying pathway structure of biochemical reaction networks, Proc. Natl. Acad. Sci. USA 95 (1998) 4193–4198.
- [30] B. M. Schmitt, The concept of “buffering” in systems and control theory: From metaphor to math, ChemBioChem 5 (2004) 1384–1392.
- [31] O. Schreier, E. Sperner, Introduction to Modern Algebra and Matrix Theory, 2nd Edition, Chelsea, 1959.
- [32] A. Schrijver, Theory of Linear and Integer Programming, John Wiley, 1998.
- [33] R. P. Stanley, Enumerative Combinatoric. Volume 1, Cambridge University Press, 1997.
- [34] T. Tian, K. Burrage, Binomial leap methods for simulating stochastic chemical kinetics, J. Chem. Phys. 121 (2004) 10356–10364.
- [35] N. G. van Kampen, Stochastic Processes in Physics and Chemistry, Elsevier, 1992.
- [36] J. S. van Zon, P. R. ten Wolde, Simulating biochemical networks at the particle level and in time and space: Green’s function reaction dynamics, Phys. Rev. Lett. 94 (2005) 128103–128107.