Reaction Cycles in Membrane Systems and Molecular Dynamics

Michael Muskulus*[1], Sanne Houweling[2], Grzegorz Rozenberg[3], Daniela Besozzi[4], Paolo Cazzaniga[5], Dario Pescini[5], Robert Brijder[3]

[1] Leiden University, Mathematical Institute
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
muskulus@math.leidenuniv.nl

[2] Vrije Universiteit Amsterdam, Faculteit der Bewegingswetenschappen
Van der Boechorststraat 9, 1081 BT Amsterdam, The Netherlands
s.houweling@fbw.vu.nl

[3] Leiden University, Leiden Institute of Advanced Computer Science
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
{rbrijder,rozenber}@liacs.nl

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

[5] Università degli Studi di Milano
Dipartimento di Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
{pescini,cazzaniga}@disco.unimib.it

Summary. We are considering molecular dynamics and (sequential) membrane systems from the viewpoint of Markov chain theory. The first step is to understand the structure of the configuration space, with respect to communicating classes. Instead of a reachability analysis by traditional methods, we use the explicit monoidal structure of this space with respect to rule applications. This leads to the notion of *precycle*, which is an element of the integer kernel of the *stoichiometric matrix*. The generators of the set of precycles can be effectively computed by an incremental algorithm due to Contejean and Devie. To arrive at a characterization of cycles, we introduce the notion of *defect*, which is a set of geometric constraints on a configuration to allow a precycle to be enabled, that is, be a cycle. An important open problem is the efficient calculation of the defects. We also discuss aspects of asymptotic behavior and connectivity, as well as give a biological example, showing the usefulness of the method for model checking.

* Corresponding author

1 Introduction

In the framework of Molecular Dynamics [9] (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 behavior can be studied. The standard algorithm used is the Stochastic Simulation Algorithm (SSA) due to Gillespie [9]. 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 [10, 28] and the efficient implementation of the reaction rate update process [8]. 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 is usually quite small).

Independently from this research, the framework of membrane systems [20, 21], for short P systems, has emerged as a well-studied model in natural computing. It features parallel, non-deterministic multiset rewriting in hierarchical reaction volumes (membranes). A special class are so-called Dynamical Probabilistic P Systems [22] (DPP) which introduce probabilities on rule application, 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 situations.

There is relatively few research on analytic properties of such systems. Chemical Reaction Network Theory [12] deals with corresponding continuous systems of ODE and is able to make strong statements about dynamic features, e.g., fixed points. Although there are some attempts into this direction [26], a similar theory for the discrete case has yet to be developed (see [19] 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 with respect to communicating classes (and a full probabilistic treatment could then be undertaken in the future). 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 [16] and on reachability issues [3, 15]. In this generality, the known algorithms are relatively inefficient. Taking into account the monoidal structure of the rewriting rules allows for much better algorithms, as well as posing some interesting mathematical questions, as we will show in the remainder of the paper.

Knowledge of molecular dynamics [9] and membrane systems [21] will be helpful for understanding the paper. In some of the examples we use a graphical representation by place-transition Petri nets [24, 17] for illustration. Relevant notions from graph theory and Markov chain theory can be found in the monographs [6, 1] and [2], respectively.

The structure of the paper is as follows: In Section 2 we provide some background information on molecular dynamics, membrane systems, Markov chain theory and directed graphs. Section 3 introduces and discusses the concepts related to reaction cycles and their calculation. In Section 4 we are concerned with communicating classes, where we focus on the asymptotic regime in which all rules can be used without constraints. A biochemical example illustrates the ideas of the paper in Section 5, and in Section 6 there is a short discussion and a few pointers to related work.

2 Background

2.1 Molecular dynamics

The standard approach to simulating a chemical system is due to Gillespie [9] 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.

2.2 Membrane systems

Membrane systems [20] (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 modeled 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 behavior, the rewriting

rules are applied in a *maximally 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 there exists no larger applicable set of rules. This can be thought of as modeling 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 textbook [21].

The *maximally parallel* execution of rules has led to some controversy [18]. It facilitates analysis, but seems unnatural with respect to biochemical reality. In fact, simulations in molecular dynamics are done sequentially for good reasons[6] 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 [22, 23] 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[7] 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 [22, 23] but are not needed for our presentation here.

[6] 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. [13]). 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.

[7] Note: In a quantum mechanical application of membrane systems (see [13] 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, 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 mapping from $I \times J \to \mathbb{R}$. 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 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 $(\mathbb{N}_0^m, +)$, called *configuration space* throughout, except for Section 4.1, 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. A topological Markov chain F_{ij} on configuration space corresponds to a map F_1 from configuration space to itself, qua

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

This map is called 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

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

We define its iterates $F_n(c)$ inductively via $F_{n+1}(c) = F_1(F_n(c))$.

Reachability of states is defined as usual: a state $c' \in \mathbb{N}_0^m$ is *(strictly) reachable* from a state $c \in \mathbb{N}_0^m$ if there exists a (positive) nonnegative integer k, such that $c' \in F_k(c)$. We denote this by $c \to c'$. The *evolution digraph* of the topological Markov chain F_{ij} on configuration space $\mathbb{N}_0^m$ is the digraph (directed graph) $G = (V, E)$, $V = \mathbb{N}_0^m$, $E = \{(v_1, v_2) \mid v_1, v_2 \in V, v_1 \to v_2\}$.

A *path* of *size* $k \in \mathbb{N}_0$ in a digraph $G = (V, E)$ is a sequence of vertices $(c_0, \ldots, c_k)$, such that $(c_i, c_{i+1}) \in E$ for all $i < k$. We also say that the path is from c_0 to c_k. In a digraph, a path $c = (c_0, \ldots, c_k)$ forms a *cycle*, if $c_0 = c_k$ and the path contains at least one edge. If all vertices of the cycle are distinct, we speak of a *simple* cycle.

Clearly, c' is reachable from c iff there is a path from c to c' in the evolution digraph. Two states $c, c' \in \mathbb{N}_0^m$ are *communicating*, if $c \to c'$ and $c' \to c$. We write this as $c \sim c'$. This is equivalent to the existence of a cycle which contains both c and c'. The communicating relation is an equivalence relation, giving rise to a partition of configuration space into *communicating classes*. The nontrivial communicating classes are also called the *strongly connected components* of the digraph. 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. The trivial communicating classes, consisting of only one element, are referred to in their totality as the *transient set* **Trans**.

We can characterize these by the cycles of the evolution digraph:

Proposition 2.1 *A cycle* $(c_0, \ldots, c_k)$ *of length* k *in the evolution digraph lies in a communicating class* $C \subseteq \mathbb{N}_0^m$*, i.e., with* $c_i \in C(c_0)$ *for all* $0 \leq i \leq k$*. Vice versa, each nontrivial communicating class* C *(with* $|C| > 1$*) 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 $[c]$.

Cycles in digraphs can be concatenated to give rise to another cycle. Concatenation is associative and the cycles form a monoid $\mathbf{Cyc} \subseteq \mathbb{N}_0^m$ under this operation. It is well known that:

Proposition 2.2 *Each cycle can be decomposed into simple cycles.*

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 path that does not leave **Trans**. We denote this by $c \sim_t c'$.

Definition 2.1 *The* reduced *configuration space, denoted by* $(\mathbb{N}_0^m)_\sim$*, is*

$$(\mathbb{N}_0^m, +)/\sim\,/\sim_t . \tag{3}$$

This space, where the (nontrivial) communicating classes are reduced to single points and the transient set is reduced to single entry/exit points, has the structure of a directed acyclic graph. Giving an initial configuration c_0, the set of all reachable states (from c_0) in $(\mathbb{N}_0^m)_\sim$ forms a tree, the so-called *reduced evolution tree.*

3 Reaction Cycles

3.1 Notation and terminology

We begin the mathematical study of sDPP systems by simplifying the notation. It is easy to show that (sequential) 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. To be more precise: There exists a bijection from one sDPP system Π to a 1sDPP system Π' such that the objects of Π are mapped bijectively to the objects of Π', the rules of Π are mapped bijectively to the rules of Π' (and such that this is compatible with the mapping on the objects), and the (initial) multisets of objects $M_0, M_2, \ldots, M_{n-1}$ are mapped to a corresponding multiset M_0'.

We therefore exclusively study 1sDPP systems, which can be seen as normal forms of sDPP systems[8]. This has the advantage that we have to deal with rewriting rules only, and no unnecessary complications due to transportation of the products is encountered. Furthermore, all finite n-sets encountered in the definition will be canonically identified with lower intervals $\{1, \ldots, n\}$ of the non-negative integers $\mathbb{Z}^+$. Thus, a multiset $c : \mathcal{O} \to \mathbb{N}_0$ over an 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 $A, B, C, \ldots$ in place of object species $1, 2, 3, \ldots$ where we see fit.

In stating the rewriting rules of the system, we allow two 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 $r : q \to p$. 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.

We do not allow *catalytic* rules of the form $M_A + C \to M_B + C$, 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*. The condition of 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 behavior 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: Instead of the (invalid) rule $A + C \to B + C$ we would consider the two rules $A + C \to D$ and $D \to B + C$, where D is a new object species that does not occur in any other rule.

Lemma 3.1 *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 $\langle p_i, q_i \rangle = p_{i,1} \cdot q_{i,1} + \cdots + p_{i,m} \cdot q_{i,m}$ 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 the necessity.

The sufficiency is trivial. □

This 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:

$$p = \max(0, r), \tag{4}$$

$$q = -\min(0, r). \tag{5}$$

[8] 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.

Definition 3.1 *A* topological molecular system *(for short* system*)* $\Pi = (m, \mathcal{R}, c_0)$ *of* type (m, n) *consists of*

- *a positive integer* m*,*
- *an* n*-tuple of nondegenerate rewriting rules* $\mathcal{R} = (r_1, \ldots, r_n)$ *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 system Π under consideration.

Note: A topological molecular system is the same as an AMR system, introduced in [18], except that in the latter we also allow inhibitory rules. Also we have 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 an easy statement of sequences of rules. Therefore the reader should keep in mind that we tacitly assume that all properties of the system do not depend on this ordering.

The integer m specifies the number of different object species of the system Π, and the multiset $c_0 \in \mathbb{N}_0^m$ is interpreted as the *initial configuration* of Π. The system evolves in discrete time by sequential application of rules, and we denote by c_t, $t = 0, 1, 2, \ldots$, the *configuration* of the system at time t, i.e., the multiset of objects in the system at that time. The *configuration space* of Π is the monoid $(\mathbb{N}_0^m, +)$, where addition is the usual vector addition. In fact, such a system is the same as a *vector addition system* [25] with an initial non-negative vector affixed to it. However, since we consider somewhat different problems here, it seems justified to use a different name.

The application of a rule $r = p - q$ is interpreted as the removal of the *reactants* $q \in \mathbb{N}_0^m$ from Π, 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 $c_0 \in \mathbb{N}_0^m$, the system evolves by sequentially applying reaction rules. These can only be applied if there are enough reactants for each rule to be consumed:

Definition 3.2 *A rule* $r_i \in \mathbb{Z}^m$ *is* enabled *in the state* $c \in \mathbb{N}_0^m$ *if*

$$r_i + c \in \mathbb{N}_0^m. \tag{6}$$

This condition is equivalent to

$$-\min(0, r_i) \leq c. \tag{7}$$

Definition 3.3 *A* trace sequence *(*trace*, for short) in* Π *is a finite sequence* $\tau = (\tau_1, \ldots, \tau_k)$*,* $\tau_i \in \{1, 2, \ldots, n\}$*.*

A trace τ of length k will be called a k-trace for short. It corresponds to a sequence of rules $r(\tau) = (r_{\tau_1}, \ldots, r_{\tau_k})$, where we use the implicitly given ordering (of the rules). In computer science this is often called a *control* sequence, but we prefer

the name trace, since we usually observe the system's behavior (in applications to biochemistry) instead of prescribing it.

Definition 3.4 *A k-trace τ is* enabled *for a configuration c_0 if each rule r_{τ_i}, $i \leq k$, is enabled in $\sum_{1 \leq j < i} r_{\tau_j} + c_0$.*

The *trajectory* $T(c_0, \tau) = (c_0, \ldots, c_k)$ of the system, given an enabled k-trace τ and an initial configuration c_0, is defined as follows:

$$c_{i+1} = c_i + R(v_{i+1} - v_i) \qquad \text{(single evolution step)}, \tag{8}$$

$$c_i = c_0 + Rv_i \qquad \text{(cumulative evolution step)}. \tag{9}$$

A topological membrane system Π corresponds to a topological Markov chain F_{ij} on configuration space, where $F_{ij} = 1$ if j is reachable from i in a single evolution step by some enabled rule, and $F_{ij} = 0$ otherwise.

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

Definition 3.5 *Given a k-trace τ, the corresponding* cumulative vector trace *is $v(\tau) = (v_0, v_1, \ldots, v_k)$ 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.*

We call $v_k(\tau)$ the *application vector* of the trace sequence τ. It counts how often each rule has been applied in the k time steps that have been traced.

3.2 The stoichiometric matrix

The notion of stoichiometric matrix [7, 26] is given in the context of membrane systems as follows:

Definition 3.6 *The* stoichiometric matrix *of the topological membrane system Π is*

$$R = \begin{pmatrix} r_1 & r_2 & \cdots & r_n \end{pmatrix}. \tag{10}$$

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.

Example 3.1 *Consider $\Pi_B = (m, \mathcal{R}, c_0)$, where $m = 3$, and $\mathcal{R} = (r_1, \ldots, r_4)$ is given by $r_1 : \lambda \to A$, $r_2 : A \to B$, $r_3 : 2A + B \to C$, $r_4 : C \to 3A$, where λ designates the empty multiset (and can be interpreted as the effect of the environment, modeling some inflow of the system). The initial configuration c_0 is left unspecified for the moment. This system has the following stoichiometric matrix:*

$$R_B = \begin{pmatrix} 1 & -1 & -2 & 3 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \tag{11}$$

This example will be further analyzed in the following sections. It is the equivalent of the Brusselator [22], a discrete model for the Belousov-Zhabotinskii chemical reaction, formulated as an open system (see Section 3.3). A Petri net representation (as a place-transition net) is given in Figure 1.

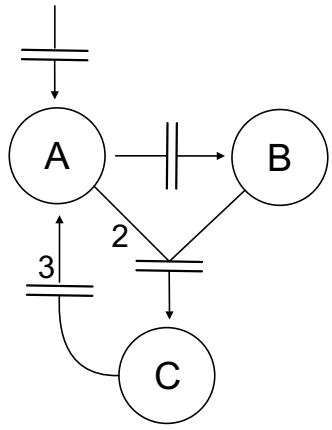

Fig. 1. Petri net representation of Brusselator example.

Note that we can decompose this 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 restrict the rules of Π to be non-degenerate.

Lemma 3.2 *A cumulative vector k-trace $v(\tau)$ is* enabled *for a configuration c_0 iff $c_0 + Rv_i \in \mathbb{N}_0^m$ for all $0 \leq i \leq k$.*

Proof. The result follows directly from the definitions. □

We are interested in the following two problems:

Problem 3.1 *Given a k-trace $\tau = (\tau_1, \ldots, \tau_k)$ in Π, characterize the set of all configurations of Π for which τ is enabled.*

Problem 3.2 *Given an application vector $a \in \mathbb{N}_0^n$, characterize* efficiently *the set of all configurations of Π for which there exists an enabled k-trace τ with $v_k(\tau) = a$.*

The first problem is easy, giving rise to the notion of *defect* (see Definition 3.9 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 Definition 3.11), but this would be too inefficient for practical use with large application vectors.

3.3 Open systems

In Example 3.1 we used a rule with the reactant multiset empty, denoted by the symbol λ on the left hand side of the rule. This was used to model some inflow of the system. In this section we show the general use of in- and out-flows.

Rules of the form $r : \lambda \to p$, $p \in \mathbb{N}_0^m$, model *inflow* of some molecular species into the system, i.e., therewith it is possible to model some nutritient/energy supplies.

Rules of the form $r : q \to \lambda$, $q \in \mathbb{N}_0^m$, 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.

Object species which partake both in the in- and outflow of the system are called *buffered* (for details of buffering in continuous modeling see [27]).

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 discussed below, 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 communication 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, and we can remove this species from all rules, i.e., replace it with λ and eliminate all redundant rules created in the process.

Criterion 3.1 *Buffered species can be removed from a system's definition. Call this the* rule of buffering.

Criterion 3.2 *Inflow or outflow species* which act only as sources or sinks, *respectively, can also be removed from the system's description. Call this the* rule of self-buffering.

Example 3.2 *The Brusselator rules as given in [22] are: Consider the system Π_B with $m = 6$ and (invalid) rules*

$$\begin{aligned} r_1 &: A \to B, \\ r_2 &: B + C \to D + E, \\ r_3 &: 2B + D \to 3B, \\ r_4 &: C \to F. \end{aligned}$$

It is usually assumed that this system has continuous inflow of species A and C, therefore we need the following additional rules $r_5 : \lambda \to A$, $r_6 : \lambda \to C$.

Furthermore we can assume continuous removal of the species E and F from the system, giving rise to additional rules $r_7 : E \to \lambda$, $r_8 : F \to \lambda$. Rewriting the third rule according to our requirement of nondegeneracy (introducing a new object species G) and reducing the system by applying Criterion 3.2 to species A, E and F, we thus arrive at the following rule set:

$$\begin{aligned} r_1' &: \lambda \to B, \\ r_2' &: B + C \to D, \\ r_3' &: 2B + D \to G, \\ r_4' &: G \to 3B, \\ r_5' &: C \to \lambda, \\ r_6' &: \lambda \to C. \end{aligned}$$

Since C is buffered*, we can apply Criterion 3.1, leaving us with*

$$\begin{aligned} r_1'' &: \lambda \to B, \\ r_2'' &: B \to D, \\ r_3'' &: 2B + D \to G, \\ r_4'' &: G \to 3B, \end{aligned}$$

which is the system studied in Example 3.1, albeit with different names of the objects.

3.4 Compound space

Sometimes it is advantageous to rewrite a system Π to an "equivalent" system Π^C which simplifies the original dynamics, but increases the number of rules. To be more precise, in this section we want to construct a system Π^C such that the system Π is *embedded* in Π^C via *injective* mappings on configurations and rules, compatible with each other (as discussed at the beginning of Section 3.1).

Definition 3.7 *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 $\mathcal{C} = \cup_{i \le n}\{\max(0, r_i), -\min(0, r_i)\}$. A compound is *trivial*, if it consists of one object species only.

Example 3.3 *Continuing Example 3.1, we have the following compounds: $c_1 = (1, 0, 0)$, $c_2 = (0, 1, 0)$, $c_3 = (0, 0, 1)$, $c_4 = (2, 1, 0)$, $c_5 = (3, 0, 0)$, corresponding to A, B, C, $2A + B$, and $3A$, respectively. The compounds c_1, c_2, and c_3 are trivial.*

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 $R^+, R^- \ge 0$, where $\langle R_i^+, R_i^- \rangle = 0$ for all i. (This correspondence is another reason why we only allow nondegenerate rules).

This idea originates in CRNT [12], 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:

The *compound space* is the nonnegative cone space generated by the compounds of the original system Π. In general, this space is smaller than the configuration space.

Proposition 3.1 *Given a system Π, its set of compounds $\mathcal{C} = \{c_1, \ldots, c_m\}$ generates all of Π'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. □

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*, the nonnegative cone generated by the union of compounds and original object species.

Example 3.4 *Consider a system Π of type $(2,1)$. The single rule is given as $r = (-2,1)$, so Π's compound set is $\mathcal{C} = \{(2,0),(0,1)\}$, and does not contain $(1,0)$. Therefore, a configuration $(2k_1+1, k_2)$, $k_1, k_2 \in \mathbb{N}_0$, cannot be mapped to a configuration in compound space.*

Given a system Π, we can then consider its (augmented) *compound system* Π^C, which is itself a topological membrane 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 3.5 *Writing Example 3.1 as an (augmented) compound system $\Pi^C = (m_C, \mathcal{R}_C, c_0)$, we find $m_C = m$ (since the compounds of Π already include all original object species), the following dynamical rules: $r_1' : \lambda \to c_1$, $r_2' : c_1 \to c_2$, $r_3' : c_4 \to c_3$, $r_4' : c_3 \to c_5$, 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:*

$$R_C = \left(\begin{array}{cccc|cccc} 1 & -1 & 0 & 0 & 2 & 3 & -2 & -3 \\ 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 \end{array}\right) \tag{12}$$

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.

The (three-dimensional) configuration space of Π is embedded in the configuration space of Π^C, and all equivalences between nontrivial compounds and their trivial constituent compounds are given by the communicating relation in Π^C.

3.5 Recurring configurations

In the theory of dynamical systems, the notions of fixed and periodic points play an important role in the analysis of a dynamical system. The analog in our discrete setting are recurring configurations.

A k-trace τ is *closed* if its corresponding application vector $v_k(\tau)$ fulfills $Rv_k(\tau) = 0$.

Definition 3.8 *A configuration $c \in \mathbb{N}_0^m$ is* recurring *if there exists an enabled closed k-trace τ.*

A recurring configuration corresponds to a *cycle* in the evolution digraph, and $v_k(\tau)$ is an element of the integer kernel $\mathrm{Ker}_{\mathbb{N}_0} R$.

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

The elements of $\mathrm{Ker}_{\mathbb{N}_0} R$, i.e., the solutions $x \in \mathbb{N}_0^n$ of $Rx = 0$, are called *reaction precycles*. They form a submonoid **PreCyc** in $\mathbb{N}_0^n$: addition of two precycles is associative and leads to another precycle. It is easy to see, by considering the *lexicographical* ordering of the precycles in $\mathbb{N}_0^n$, that this submonoid is *finitely* generated by its *minimal* (with respect to the inclusion ordering) elements [4].

If enough objects are present in a configuration to make a reaction precycle possible (see the notion of *defect* in Definition 3.9), it is called a *reaction cycle* for this configuration.

3.6 The incremental algorithm of Contejean and Devie

Finding the (minimal) reaction precycles corresponds to the problem of solving a linear homogenous diophantine equation. Background information and an effective algorithm for the determination of the minimal precycles can be found in [5], building on earlier work of Fortenbacher [4] and others.

A short sketch of the basic algorithm follows. Consider the pseudocode in Table 1. Correctness and termination are proved in [5]. The algorithm starts with the unit step vectors, one for each rule, as starting elements (tentative solutions). In each iterative step it checks if the elements solve the equation $Ax = 0$. 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 lexicographical ordering) any minimal element are removed, since they are not minimal. All 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).

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

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

Table 1. The incremental algorithm of Contejean & Devie

Example 3.6 *Applied to the Brusselator example, the above algorithm finds one minimal precycle* $x_{min} = (0, 1, 1, 1)$. *This is the (only) reaction precycle of the system and can occur, for example, as the following trajectory:* $3A \to 2A + B \to C \to 3A$.

3.7 Defects and their calculation

We introduce the following simple, but important concept:

Definition 3.9 *The* defect *of a closed* k*-trace* τ *is*

$$\mathrm{def}(\tau) = -\min\{0, Rv_1(\tau), Rv_2(\tau), \dots, Rv_k(\tau) = 0\} \in \mathbb{N}_0^m. \tag{13}$$

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 $d = \mathrm{def}(\tau)$ contains the minimum number of objects of species i needed in a configuration to make the trace τ possible, i.e., enabled. See Figure 2 for the geometrical intuition behind this.

To define the defect for an application vector (of a reaction precycle), we need to consider all possible permutations of the application of its rules. Formally this is done as follows:

Definition 3.10 *Given a multiset $x \in \mathbb{N}_0^n$ of size $|x| = \sum_i x_i$, a* multiset ordering *σ of x is a sequence $(\sigma_1, \ldots, \sigma_{|x|})$, with $0 \leq \sigma_i \leq n$, such that*

$$\sum_{1 \leq i \leq n} \delta^j_{\sigma_i} = x_j$$

for all $j \leq n$, where δ^j_i denotes the Kronecker delta, which is zero if $i \neq j$, and has value one otherwise.

The set of all multiset orderings of a multiset x is denoted by $\Xi(x)$. It contains

$$|\Xi(x)| = \binom{|x|}{|x_1|, |x_2|, \ldots, |x_n|} = \frac{|x|!}{|x_1|! \cdot |x_2|! \cdots |x_n|!}$$

elements.

Example 3.7 *Given $x_{min} = (0, 1, 1, 1)$ from Example 3.6, $\Xi(x_{min}) \approx S_3$, the set of all permutations of three elements. Given $x = (2, 1, 2)$ we get $\Xi(x) = \{(1, 1, 2, 3, 3), (1, 2, 1, 3, 3), \ldots\}$, $|\Xi(x)| = 30$.*

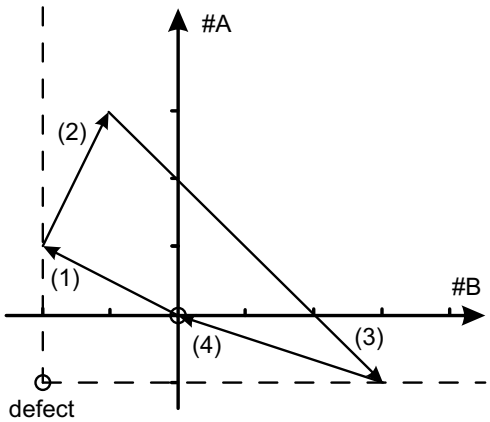

Fig. 2. Example illustrating the meaning of *defect*. The configuration space is two-dimensional, and the reaction precycle shown results from the application of rules $2B \to A$, $B \to 2A$, $4A \to 3B$, $3B \to A$. The defect of this particular 4-trace is $(1, 2)$.

From the definitions we see that a multiset ordering $\sigma \in \Xi(x)$ is a $|x|$-trace. The *inclusion* ordering of $\mathbb{N}_0^n$ is given for two configurations $c, c' \in \mathbb{N}_0^n$ by $c \leq c'$ iff $c' - c \in \mathbb{N}_0^n$.

Definition 3.11 *Given a reaction precycle $x \in \mathbb{N}_0^n$, 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:*

$$\mathbf{def}(x) = \textit{ min. elements}\{\mathrm{def}(\sigma) \mid \sigma \in \Xi(x)\}. \tag{14}$$

Example 3.8 *The precycle x_{min} from Example 3.6 has a defect consisting of three elements:*

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

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

Definition 3.12 *The* defect space *of a reaction precycle $x \in \mathbb{N}_0^n$ is the* cone $\mathrm{ds}(x) = \mathbf{def}(x) + \mathbb{N}_0^m = \{v_1 + v_2 \mid v_1 \in \mathbf{def}(x), v_2 \in \mathbb{N}_0^m\}$.

Note: Addition refers here to the monoid operation. The defect space can also be characterized by

$$\mathrm{ds}(x) = \{c \in \mathbb{N}_0^m \mid c \geq y, \text{ for some } y \in \mathbf{def}(x)\} \tag{15}$$

Geometrically, it is the region of the configuration space in which there exists some (multiset) ordering $\sigma \in \Xi(x)$ of (the rules from) x, such that the system can cycle back to where it was before using the rules in the order σ. 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 a (nonnegative, integer) solution of the linear equation $Rx = 0$, this is a nonnegative, integer, linear combination

$$x = \sum_{\substack{1 \leq i \leq |\mathbf{def}(x)|, \\ y_i \in \mathbf{def}(x)}} a_i y_i,\ a_i \in \mathbb{N}_0 \tag{16}$$

of minimal precycles.

Proposition 3.2 *Given an arbitrary precycle $x \in \mathbb{N}_0^n$, we have*

$$\mathrm{ds}(x) \supseteq \bigcap_{y \in \mathbf{def}(x)} \mathrm{ds}(y). \tag{17}$$

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

The following question is of central interest:

Problem 3.3 *Given an arbitrary precycle $x \in \mathbb{N}_0^n$, is it true that*

$$\mathrm{ds}(x) = \bigcap_{y \in \mathbf{def}(x)} \mathrm{ds}(y)\ ? \tag{18}$$

Note: If this would be the case, 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

$$\mathrm{ds}(x_1 + x_2) = \{c \in \mathbb{N}_0^m \mid c \geq \max(y_1, y_2), \text{ for some } y_i \in \mathbf{def}(x_i), i = 1, 2\}, \quad (19)$$

such that

$$\mathbf{def}(x_1 + x_2) = \bigcup_{\substack{y_i \in \mathbf{def}(x_i), \\ i=1,2}} \max(y_1, y_2) \quad (20)$$

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

4 Communicating Classes

4.1 Asymptotic cycles

In this section we want to consider different questions. Call $(\mathbb{Z}^m, +)$ *asymptotic space* and consider the behavior of Π 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 Π in the *asymptotic 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 precycles are cycles. The notions from Section 2.4, formulated for $(\mathbb{N}_0^m, +)$ originally, carry over to the space $(\mathbb{Z}^m, +)$ easily.

Problem 4.1 *How many communicating classes exist in the asymptotic 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 asymptotic lattice of configurations is translation-invariant) and thus there exists an infinite number of asymptotically different communicating classes. Looking at the reduced (asymptotic) configuration space (recall Definition 2.1), it will be an infinite tree.

A rule is called *transient*, if it is not part of any precycle. Otherwise it 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.

Since cycles are asymptotically equivalent to precycles, and all cycles are generated by the minimal precycles, we only need to check this condition on the latter.

The set of rules $\mathcal{R}$ is called *reversible*, if it contains no transient rules. Otherwise it is called *transient*. A reversible set of rules guarantees that the communicating classes in the evolution digraph are independent. 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, and a small

perturbation of the system (e.g., addition or removal of one object of some species) could lead the system from one such component to another.

For the corresponding Markov chain this would mean that there would be an infinite number of stationary distributions, so it is an important to know under which conditions this happens.

Definition 4.1 *The* compacted asymptotic configuration space *of a topological membrane system Π is the quotient $\mathbb{Z}^m/\sim$.*

Problem 4.2 *Under which conditions is the compacted asymptotic configuration space of a system Π finite?*

The following is a sufficient condition for finiteness:

Proposition 4.1 *Compacted asymptotic configuration space $\mathbb{Z}^m/\sim$ is trivial (i.e.,consists of only one communicating class) iff the origin is equivalent (under the communication relation) to all unit vectors e_i, $1 \leq i \leq m$ and their negative counterparts $-e_i$, $1 \leq i \leq m$.*

Proof. (Similar to the proof of Proposition 3.1.)
Necessity is shown first: If the unit vectors and their negative copies are equivalent to the origin, this means they are reachable by cycles in asymptotic space (by applying rules). Since asymptotic space is homogeneous, and therefore translation-invariant, we can move the origin to one of these vectors. Iterating this argument shows that we can reach all of asymptotic space, therefore it becomes trivial under the reachability equivalence relation.

Sufficiency is immediate, since the unit vectors and their counterparts trivially are elements of asymptotic space. □

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 [5]:

Introduce an extra variable $x_0 \in \mathbb{N}_0$ with $r_0 = \mp e_i$ as new first row of the otherwise identical stoichiometric matrix R (note the change of sign). In the algorithm in Table 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.

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 [28]. Biological species names have been replaced by capital letters. The correspondence for the most important reactants has been given in Table 2.

$$\mathbf{A} + \mathbf{B} \longrightarrow \mathbf{C} \tag{21}$$
$$\mathbf{C} \longrightarrow \mathbf{A} + \mathbf{B} \tag{22}$$
$$\mathbf{C} \longrightarrow \mathbf{D} \tag{23}$$
$$\mathbf{D} \longrightarrow \mathbf{A} + \mathbf{E} + \mathbf{F} \tag{24}$$
$$\mathbf{F} \longrightarrow \mathbf{G} \tag{25}$$
$$\mathbf{G} \longrightarrow \mathbf{H} + \mathbf{I} \tag{26}$$
$$\mathbf{I} \longrightarrow \mathbf{B} \tag{27}$$
$$\mathbf{E} + \mathbf{J} \longrightarrow \mathbf{K} \tag{28}$$
$$\mathbf{H} + \mathbf{J} \longrightarrow \mathbf{L} \tag{29}$$
$$\mathbf{K} \longrightarrow \mathbf{E} + \mathbf{J} \tag{30}$$
$$\mathbf{L} \longrightarrow \mathbf{J} + \mathbf{H} \tag{31}$$
$$\mathbf{K} \longrightarrow \mathbf{E} + \mathbf{M} \tag{32}$$
$$\mathbf{L} \longrightarrow \mathbf{H} + \mathbf{N} \tag{33}$$
$$\mathbf{M} \longrightarrow \mathbf{O} \tag{34}$$
$$\mathbf{N} \longrightarrow \mathbf{P} \tag{35}$$
$$\mathbf{O} \longrightarrow [\ \mathbf{R}\ ] \tag{36}$$
$$\mathbf{P} \longrightarrow [\ \mathbf{S}\ ] \tag{37}$$
$$\mathbf{E} \longrightarrow [\ \mathbf{T}\ ] \tag{38}$$
$$\mathbf{H} \longrightarrow [\ \mathbf{X}\ ] \tag{39}$$
$$\mathbf{O} + \mathbf{U} \longrightarrow \mathbf{V} \tag{40}$$
$$\mathbf{V} \longrightarrow \mathbf{O} + [\ \mathbf{W}\ ] \tag{41}$$
$$\mathbf{P} \longrightarrow \mathbf{U} + \mathbf{Q} \tag{42}$$
$$\mathbf{Q} \longrightarrow \mathbf{P} \tag{43}$$

More information on the (biological) meaning of these objects and stochastic simulation results can be also found in [28]. In the equations we have already indicated by bracketing the outflows and introduced an extra variable Q to ensure non-degeneracy. In our version of the model the outflows will be replaced by rules of the form $O \to \lambda, P \to \lambda$, etc.

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. These consist of molecules as well as membranes (e.g., Ribosome). The latter are treated in the same manner as normal molecules, in a slight generalization of Molecular Dynamics.

Without these outflows, the system will only show the following three internal cycles (which are of interest of their own, of course), given in some arbitrary trace sequence:

$$\mathbf{A}+\mathbf{B} \rightarrow \mathbf{C} \rightarrow \mathbf{A}+\mathbf{B} \tag{44}$$

$$\mathbf{E}+\mathbf{J} \rightarrow \mathbf{K} \rightarrow \mathbf{E}+\mathbf{J} \tag{45}$$

$$\mathbf{H}+\mathbf{J} \rightarrow \mathbf{L} \rightarrow \mathbf{H}+\mathbf{J} \tag{46}$$

Figure 3 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. It is nontrivial to find the other precycles for this model.

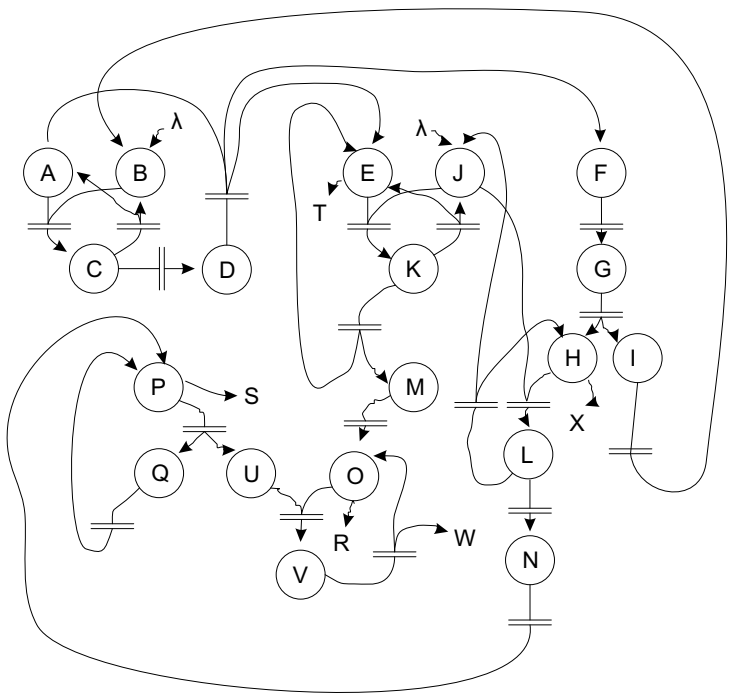

Fig. 3. Petri net representation of LacZ/LacY example. Outflow species and additional inflows (see text) are shown without places.

Applying the above algorithm gives us the following additional possibilities[9]:

[9] Note: Most reaction cycles in biology are rather small, given the constant threat of turnover, i.e., spontaneous degradation of biologically active molecules.

$$\begin{aligned}\mathbf{P}+\mathbf{O}+\mathbf{U} \;\to\; \mathbf{P}+\mathbf{V} \;\to\; \mathbf{P}+\mathbf{O}+\mathbf{W} \;\to\; \\ \to\; \mathbf{P}+\mathbf{O} \;\to\; \mathbf{Q}+\mathbf{U}+\mathbf{O} \;\to\; \\ \to\; \mathbf{P}+\mathbf{U}+\mathbf{O}\end{aligned} \tag{47}$$

$$\begin{aligned}\mathbf{A}+\mathbf{B} \;\to\; \mathbf{C} \;\to\; \mathbf{D} \;\to\; \mathbf{A}+\mathbf{E}+\mathbf{F} \;\to\; \\ \to\; \mathbf{A}+\mathbf{E}+\mathbf{G} \;\to\; \mathbf{A}+\mathbf{E}+\mathbf{H}+\mathbf{I} \;\to\; \\ \to\; \mathbf{A}+\mathbf{E}+\mathbf{H}+\mathbf{B} \;\to\; \mathbf{A}+\mathbf{H}+\mathbf{B} \;\to\; \mathbf{A}+\mathbf{B}\end{aligned} \tag{48}$$

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 we need to extend the model. We have to include inflows for species B and J, representing RNAP and Ribosome membrane (site) entities that can be used up. With two additional rules $\lambda \to B$, $\lambda \to J$ we find the following additional solutions:

$$\mathbf{E}+\mathbf{J} \;\to\; \mathbf{K} \;\to\; \mathbf{E}+\mathbf{M} \;\to\; \mathbf{E}+\mathbf{O} \;\to\; \mathbf{E} \;\to\; \mathbf{E}+\mathbf{J} \tag{49}$$

$$\mathbf{H}+\mathbf{J} \;\to\; \mathbf{L} \;\to\; \mathbf{H}+\mathbf{N} \;\to\; \mathbf{H}+\mathbf{P} \;\to\; \mathbf{H} \;\to\; \mathbf{H}+\mathbf{J} \tag{50}$$

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

Now all the rules, apart from the one for the inflow of species B, are used in some cycle.

Criterion 5.1 *Assume (for biological systems) that each inflow and outflow species is part of some further reaction cycle (in a larger system). In the triviality check according to Proposition 4.1 we therefore only need to check the existence of a solution $x \in \mathbb{N}_0^m$, $Rx = e_i$, for all $i \leq m$ which do not correspond to one such inflow/outflow object. Call this is the* rule of (biological) recycling.

Applying this criterion and running the modified algorithm gives no solution for $e_{\{Q=19\}} = 1$ (as well as to some other unit vectors). This alone shows that the system will still be transient, with an infinite number of communicating classes that are different by the amount of species Q — or species P, for that matter, which will be produced out of the intermediary state Q eventually. In simulations it is therefore possible to use one constant initial value of these species (asymptotically, i.e., disregarding possible effects of the defects).

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 internal as well as production/degradation cycles and had to expand the model appropriately to find further important cycles. 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. Furthermore we have given some open problems concerning the geometry of communicating classes.

We also should mention the related work of [11] which is concerned with finding similar cycle bases for chemical reaction networks, and the interesting aspect of decompositions based on cycles [14].

Acknowledgments. M. Muskulus and R. Brijder acknowledge support by the Nederlandse Wetenschappelijke Organisatie NWO under grant no. 635.100.006. The work of D. Besozzi has been supported by the European Research Training Network "Segravis".

References

1. J. Bang-Jensen, G. Gutin: *Digraphs. Theory, Algorithms and Applications.* Springer, 2002.
2. P. Brémaud: *Markov Chains. Gibbs Fields, Monte Carlo Simulation, and Queues.* Vol. 31 of Texts in Applied Mathematics, Springer, 1999.
3. P. Buchholz, P. Kemper: Hierarchical reachability graph generation for Petri nets. *Formal Methods in System Design*, 21 (2002), 281–315.
4. M. Clausen, A. Fortenbacher: Efficient solution of linear diophantine equations. *J. Symbolic Computation*, 8 (1989), 201–216.
5. E. Contejean, H. Devie: An efficient incremental algorithm for solving systems of linear diophantine equations. *Information and Computation*, 113 (1994), 143–172.
6. R. Diestel: *Graph Theory.* 3rd Edition, Springer, 2005.
7. P. Dittrich, P.S. di Fenizio: *Chemical Organization Theory: Towards a Theory of Constructive Dynamical Systems.* Tech. rep., arXiv preprint: `http://xxx.lanl.gov/q-bio.MN/abs/0501016` (2005).
8. 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.
9. D.T. Gillespie: Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81 (1977), 2340–2361.
10. D.T. Gillespie: Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115 (2001), 1716–1733.
11. P.M. Gleiss, P.F. Stadler, A. Wagner, D.A. Fell: Relevant cycles in chemical reaction networks. *Adv. Complex Systems*, 4, 5 (2001), 207–226.
12. J. Gunawardena: *Chemical Reaction Network Theory for In–silico Biologists.* Tech. rep., Bauer Center for Genomics Research, Harvard University (June 2003).

13. A. Leporati, G. Mauri, C. Zandron: Quantum sequential P systems with unit rules and energy assigned to membranes. In *Membrane Computing, 6th International Workshop, WMC 2005* (R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), LNCS 3850, Springer, 310–325.
14. M. Loebl, M. Matamala: Some remarks on cycles in graphs and digraphs. *Discrete Mathematics*, 233, 1-3 (2001), 175–182.
15. 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, 238–246.
16. A. Miner, D. Parker: Symbolic representation and analysis of large probabilistic systems. In *Validation of Stochastic Systems* (C. Baier, B.R. Haverkort, H. Hermanns, J.-P. Katoen, M. Siegle, eds.), LNCS 2925, Springer, 2004, 296–338.
17. T. Murata: Petri nets: Properties, analysis and applications. *Proc. Inst. Electr. Eng.*, 77, 4 (1989), 541–580.
18. M. Muskulus, R. Brijder: First steps toward a geometry of computation. In *Third Brainstorming Week on Membrane Computing* (M.A. Gutierrez-Naranjo, A. Riscos-Nunez, F.J. Romero-Campero, eds.), Fenix Editora, Sevilla, 2005, 197–218.
19. M. Muskulus, R. Brijder: Complexity of biocomputation: Symbolic dynamics in membrane systems. *Int. J. Found. Comp. Sci.*, 17, 1 (2006), 147–165.
20. Gh. Păun: Computing with membranes. *J. Comput. Syst. Sci.*, 61, 1 (2000), 108–143.
21. Gh. Păun: *Membrane Computing: An Introduction.* Springer, 2002.
22. D. Pescini, D. Besozzi, G. Mauri, C. Zandron: Dynamical probabilistic P systems. *Int. J. Found. Comp. Sci.*, 17, 1 (2006), 183–204.
23. D. Pescini, D. Besozzi, C. Zandron, G. Mauri: Analysis and simulation of dynamics in probabilistic P systems. In *Pre-Proceedings of the 11th International Conference on DNA Computing* (A. Carbone, M. Daley, L. Kari, I. McQuillan, N. Pierce, eds.), London, Ontario, Canada, 2005, 310–321.
24. J.L. Peterson: *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, 1981.
25. C. Reutenauer: *Aspects mathématiques des Résaux de Pétri.* Masson, 1989.
26. C.H. Schilling, B.O. Palsson: The underlying pathway structure of biochemical reaction networks. *Proc. Natl. Acad. Sci. USA*, 95 (1998), 4193–4198.
27. B.M. Schmitt: The concept of "buffering" in systems and control theory: From metaphor to math. *ChemBioChem*, 5 (2004), 1384–1392.
28. T. Tian, K. Burrage: Binomial leap methods for simulating stochastic chemical kinetics. *J. Chem. Phys.*, 121, 21 (2004), 10356–10364.