COMPUTING SCIENCE

Membrane Systems with Qualitative Evolution Rules

Jetty Kleijn and Maciej Koutny

TECHNICAL REPORT SERIES

No. CS-TR-1287 October 2011

TECHNICAL REPORT SERIES

No. CS-TR-1287 October, 2011

Membrane Systems with Qualitative Evolution Rules

J. Kleijn, M. Koutny

Abstract

In membrane systems, biochemical reactions taking place in the compartments of a cell are abstracted to evolution rules that specify which and how many objects are consumed and produced. The recently proposed reaction systems also investigate processes carried by biochemical reactions, but the resulting computational model is remarkably different. A key difference is that in reaction systems, biochemical reactions are modeled using a qualitative rather than a quantitative approach.
In this paper, we introduce so-called set membrane systems, a variant of membrane systems with qualitative evolution rules inspired by reaction systems. We then relate set membrane systems to Petri nets which leads to a new class of Petri nets: set-nets with localities. This Petri net model provides a faithful match with the operational semantics of set membrane systems.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

KLEIJN, J., KOUTNY, M.

Membrane Systems with Qualitative Evolution Rules
[By] J. Kleijn, M. Koutny
Newcastle upon Tyne: Newcastle University: Computing Science, 2011.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1287)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1287

Abstract

In membrane systems, biochemical reactions taking place in the compartments of a cell are abstracted to evolution rules that specify which and how many objects are consumed and produced. The recently proposed reaction systems also investigate processes carried by biochemical reactions, but the resulting computational model is remarkably different. A key difference is that in reaction systems, biochemical reactions are modeled using a qualitative rather than a quantitative approach. In this paper, we introduce so-called set membrane systems, a variant of membrane systems with qualitative evolution rules inspired by reaction systems. We then relate set membrane systems to Petri nets which leads to a new class of Petri nets: set-nets with localities. This Petri net model provides a faithful match with the operational semantics of set membrane systems.

About the authors

Jetty Kleijn is a visiting fellow within the School of Computing Science, Newcastle University.

Maciej Koutny obtained his MSc (1982) and PhD (1984) from the Warsaw University of Technology. In 1985 he joined the then Computing Laboratory of the University of Newcastle upon Tyne to work as a Research Associate. In 1986 he became a Lecturer in Computing Science at Newcastle, and in 1994 was promoted to an established Readership at Newcastle. In 2000 he became a Professor of Computing Science.

Suggested keywords

MEMBRANE SYSTEM
REACTION SYSTEM
BIOCHEMISTRY
NATURAL COMPUTING
PETRI NET
SET-NET
LOCALITY
INHIBITOR
PROMOTER

Membrane Systems with Qualitative Evolution Rules

Jetty Kleijn[1] and Maciej Koutny[2]

[1] LIACS, Leiden University
P.O.Box 9512, NL-2300 RA Leiden, The Netherlands
kleijn@liacs.nl

[2] School of Computing Science, Newcastle University
Newcastle upon Tyne, NE1 7RU, United Kingdom
maciej.koutny@ncl.ac.uk

Abstract. In membrane systems, biochemical reactions taking place in the compartments of a cell are abstracted to evolution rules that specify which and how many objects are consumed and produced. The recently proposed reaction systems also investigate processes carried by biochemical reactions, but the resulting computational model is remarkably different. A key difference is that in reaction systems, biochemical reactions are modeled using a qualitative rather than a quantitative approach.
In this paper, we introduce so-called set membrane systems, a variant of membrane systems with qualitative evolution rules inspired by reaction systems. We then relate set membrane systems to Petri nets which leads to a new class of Petri nets: set-nets with localities. This Petri net model provides a faithful match with the operational semantics of set membrane systems.
Keywords: membrane system, reaction system, biochemistry, natural computing, Petri net, set-net, locality, inhibitor, promoter.

1 Introduction

Membrane systems, or P systems ([12–15, 19]) are a computational model inspired by the compartmentisation of living cells and the biochemical reactions taking place in such compartments. These reactions are abstracted to evolution rules specifying which and how many new objects (molecules) can be produced from objects of a certain kind and quantity, possibly involving a transfer to a neighbouring compartment. The dynamic aspects of a membrane system and its potential behaviour (its computations) derive from these evolution rules. When a membrane system evolves, the current state of any given compartment is represented as a *multiset* of objects, and each computational action is represented as a *multiset* of simultaneously executed (multiple copies of) individual evolution rules. Such strong reliance on counting (through multiple copies of objects and rules) may lead to potential problems in two respects. First, one may wonder how realistic is the counting (multiset) mechanism if one needs to represent huge

numbers of molecules and instances of biochemical reactions. Second, a membrane system would normally have an infinite state space, making the application of formal verification techniques impractical or indeed impossible (there exists a rich body of results proving Turing completeness of even very simple kinds of membrane systems).

A radical solution to the state space problems can be provided by reaction systems [2–4] which are also a formal framework for the investigation of processes carried by biochemical reactions. Reaction systems, however, model biochemical reactions in living cells using *qualitative* — based on presence and absence of entities — rather than *quantitative* term rewriting rules. Hence the semantical model of reaction systems is remarkably different from those underlying other existing models of computation, including membrane systems. Moreover, the state of a (sub)system can be represented by a *set* rather than a multiset of objects, which leads to state spaces that are always finite. Further fundamental differences between membrane systems and reaction systems are the compartmentalization present in the former, including the possibility of dynamically changing structure of the membranes. Another one is the non-persistence of objects in reaction systems, i.e., an object which is not sustained by executed rules is removed from the system. Also, each rule of a reaction system specifies an inhibition set. It is important to note that reaction systems are a formal model for the investigation and understanding of interactions between biochemical reactions in living cells, leading to an abstract theory of the resulting dynamic processes.

The first aim of this paper is to exploit the qualitative approach to modelling biochemistry embodied by reaction systems in the realm of membrane systems. The second aim is to build bridges allowing one to import analytical tools from more established models and approaches the domain of the new model.

We will address the first aim by defining the *set membrane systems* model which is a qualitative variation of the standard quantitative membrane systems with evolution rules and execution semantics inspired by reaction systems. In a nutshell, in set membrane systems, all modelling devices as well as execution rules will be based on sets (of objects or rules) together with the associated set theoretic operations, rather than on multisets and multiset operations. This is similar to the operation of the membrane systems discussed in [1], where the quantitative approach was used when sending objects to the external environment, and the qualitative one was used in the application of rules within the membranes.

The second of our aims will be addressed by providing a faithful model translation from set membrane systems to a class of Petri nets. Petri nets are an operational model for concurrent systems with distributed states and actions with local causes and effects. In Petri nets, such as the classical Place-Transition nets (PT-nets), resources and actions are represented in a quantitative way, essentially as in the standard membrane systems. This was in part the reason why in previous work [8, 10] we were able to give membrane systems a Petri net semantics, through an extension of PT-nets with a concept of transition *locality* used to reflect the compartmentisation of a membrane system. In another strand of

our work, we introduced in [11] a new class of Petri nets, called set-nets, as a net based computational model matching very closely that exhibited by reaction systems. In this paper, we will combine the ideas contained in [8, 10, 11] and introduce a new model of set-nets with localities which provides a behavioural match for the set membrane systems.

The paper is organised in the following way. In Section 2 we formalise the basic ideas concerning qualitative membrane systems and in Section 3, we introduce the new class of nets corresponding to basic set membrane systems. The details of the translation from set membrane systems to nets are presented in Section 4. Finally, Section 5 explains how to introduce promoters and inhibitors to basic set membrane systems, and then how to model these features in the Petri net domain.

2 Basic set membrane systems

In this section, we introduce a simple class of qualitative membrane systems. The presentation follows in many respects the standard approach to defining membrane systems. The key difference is the 'qualitative' rather than 'quantitative' application of evolution rules to change the current state of a system.

A *membrane structure* μ (of degree $m \geq 1$) is given by a rooted tree with m nodes identified with the integers $1, \ldots, m$. We will write $(i, j) \in \mu$ or $i = \mathit{parent}(j)$ to mean that there is an edge from i (parent) to j (child) in the tree of μ, and $i \in \mu$ to mean that i is a node of μ. The nodes of a membrane structure represent nested membranes which in turn determine compartments. Compartment j is enclosed by membrane j and lies in-between j and its children (if any). Figure 1 shows a membrane structure (with $m = 5$) together with the corresponding compartments. Note that 1 is the root node, $(1, 2) \in \mu$ and $3 = \mathit{parent}(5)$.

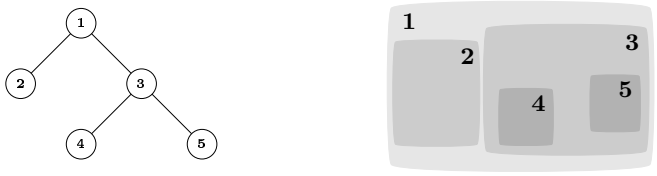


Fig. 1. A membrane structure and its compartments.

Let V be a finite alphabet of *objects*. A *basic set membrane system* over the membrane structure μ is a tuple $\Sigma = (V, \mu, w_1^0, \ldots, w_m^0, R_1, \ldots, R_m)$ such that, for every membrane i of μ, $w_i^0 \subseteq V$ is a set of objects, and R_i is a finite set of *evolution rules*. Each evolution rule $r \in R_i$ is of the form $\mathit{lhs}^r \rightarrow \mathit{rhs}^r$, where

$lhs^r \subseteq V$ is a non-empty set of objects, and $rhs^r \subseteq V_i$ is a set of (indexed) objects, with V_i being defined as:

$$V_i = V \cup \{a_{out} \mid a \in V\} \cup \{a_{in_j} \mid a \in V \text{ and } (i,j) \in \mu\} .$$

It is assumed that if i is the root of μ then no indexed object of the form a_{out} belongs to rhs^r. [1]

The tuple $C_0 = (w_1^0, \ldots, w_m^0)$ is the *initial configuration* (or initial state) of Σ. In general, a *configuration* of Σ is a tuple $C = (w_1, \ldots, w_m)$ of sets of objects. Below we assume that Σ is a fixed basic set reaction system.

We refer to lhs^r as the left hand side of the rule r, and rhs^r as its right hand side. lhs^r specifies which objects are needed as input for an execution of this rule, and rhs^r specifies which new objects are produced and where they are deposited. An indexed object $a_{in_j} \in rhs^r$ indicates that a newly produced object a is sent to a child node (compartment) j, and a_{out} indicates that a is sent to the parent node. If no index is present, the newly produced object remains in the same compartment. Figure 2 depicts a basic set membrane system over the membrane structure μ shown in Figure 1. Note that $V = \{a, b, c\}$, $lhs^{r_{21}} = \{a, c\}$, $rhs^{r_{12}} = \{b, c_{in_2}, a_{in_3}\}$, $w_1^0 = \{a, b\}$ and $w_5^0 = \varnothing$.

As a consequence of the execution of evolution rules as outlined above, a set membrane system evolves from configuration to configuration. There are different ways to combine evolution rules (see e.g., [6]). We distinguish four main *execution modes*, all expressed through the notion of a vector set-rule.

A *vector set-rule* of Σ is a tuple $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$ where, for each membrane i of μ, $\mathbf{r}_i$ is a set of rules from R_i. For two vector set-rules, $\mathbf{r}$ and $\mathbf{r}'$, we denote $\mathbf{r} \subseteq \mathbf{r}'$ if $\mathbf{r}_i \subseteq \mathbf{r}'_i$, for each $i \leq m$; and $\mathbf{r} \subset \mathbf{r}'$ if $\mathbf{r} \subseteq \mathbf{r}'$ and $\mathbf{r} \neq \mathbf{r}'$. We also lift the notion of left and right hand sides of rules to sets of rules in vector set-rules. For a vector set-rule $\mathbf{r}$ and $i \leq m$, we respectively denote by:

$$lhs_i^{\mathbf{r}} = \bigcup_{r \in \mathbf{r}_i} lhs^r \quad \text{and} \quad rhs_i^{\mathbf{r}} = \bigcup_{r \in \mathbf{r}_i} rhs^r$$

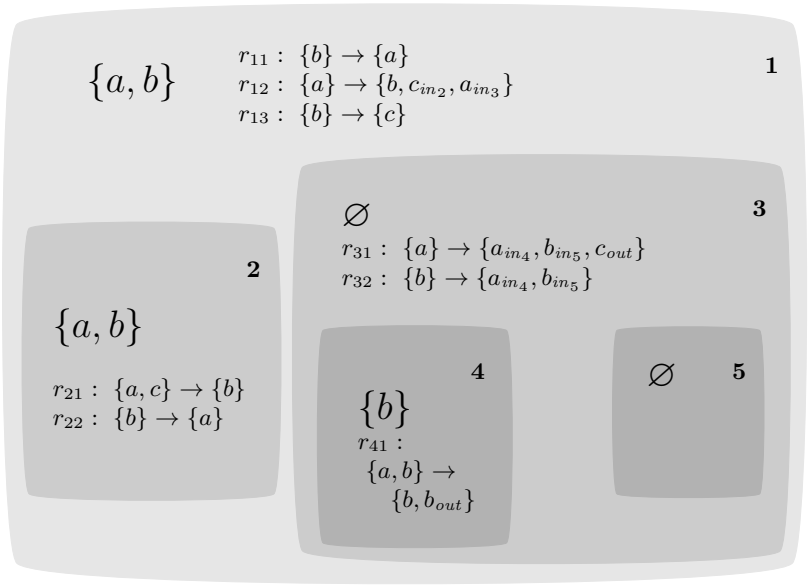
the set of all the objects in the left hand sides of the rules in $\mathbf{r}_i$, and the set of all the (indexed) objects in their right hand sides. Intuitively, $lhs_i^{\mathbf{r}}$ specifies the objects needed for the execution of the evolution rules in $\mathbf{r}_i$.
We then say that a vector set-rule $\mathbf{r}$ is:

- *free-enabled* at a configuration $C = (w_1, \ldots, w_m)$ if $lhs_i^{\mathbf{r}} \subseteq w_i$, for each i.

Moreover, a free-enabled $\mathbf{r}$ is:

- *min-enabled* if $|\mathbf{r}_1| + \cdots + |\mathbf{r}_m| = 1$;

[1] In other words, objects sent out to the environment are not relevant anymore, they do not come back [13]. Note that if it is necessary to send concrete objects to the external environment as in [1], one can easily introduce another root membrane to model this environment as an encompassing compartment.



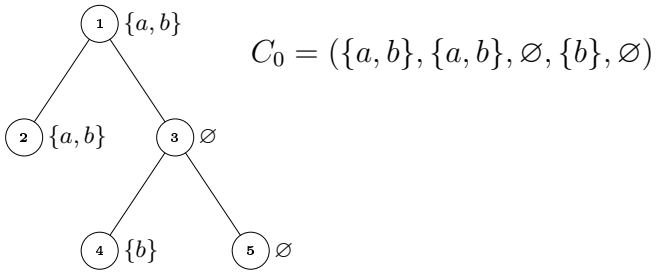


Fig. 2. A basic membrane system over the membrane structure μ shown in Figure 1. Its initial configuration is shown explicitly underneath using μ with each set w_i^0 placed next to the corresponding node i.

- *max-enabled* if no $\mathbf{r}_i$ can be extended to yield a vector set-rule which is free-enabled at C (i.e., there is no free-enabled vector set-rule $\mathbf{r}'$ such that $\mathbf{r} \subset \mathbf{r}'$); and
- *lmax-enabled* if no non-empty $\mathbf{r}_i$ can be extended to yield a vector set-rule which is free-enabled at C (i.e., there is no free-enabled vector set-rule $\mathbf{r}'$ such that $\mathbf{r} \subset \mathbf{r}'$ and $\mathbf{r}'_i = \varnothing$, whenever $\mathbf{r}_i = \varnothing$).

For the initial configuration of the running example, we have:

- $\langle \varnothing, \{r_{21}\}, \varnothing, \varnothing, \varnothing \rangle$ is not free-enabled;
- $\langle \{r_{11}\}, \varnothing, \varnothing, \varnothing, \varnothing \rangle$ is min-enabled but not lmax-enabled;
- $\langle \{r_{11}, r_{12}, r_{13}\}, \varnothing, \varnothing, \varnothing, \varnothing \rangle$ is lmax-enabled but not max-enabled; and

– $\langle\{r_{11}, r_{12}, r_{13}\}, \{r_{22}\}, \varnothing, \varnothing, \varnothing\rangle$ is max-enabled.

If $\mathbf{r}$ is free-enabled (*free*) at a configuration C, then each membrane i contains all kinds of objects needed for the execution of the evolution rules in $\mathbf{r}_i$; it is worth pointing out that a particular (kind of) object can be used as as input to different rules in $\mathbf{r}_i$. Maximal enabledness (*max*) of $\mathbf{r}$ requires that any extra rule demands the presence of objects that C does not provide. Note that there is always exactly one max-enabled vector set-rule. Locally maximal enabledness (*lmax*) is similar but in this case only those compartments that are actually involved in $\mathbf{r}$ do not enable any other rules; in other words, each compartment either uses no rule, or uses all free-enabled rules. Minimal enabling (*min*) allows only a single rule to be applied at any time. We next describe the effect of the execution of the rules for any mode of execution $\mathfrak{m} \in \{free, min, max, lmax\}$.

We say that a configuration $C = (w_1, \ldots w_m)$ can $\mathfrak{m}$-*evolve* by a vector set-rule $\mathbf{r}$ which is $\mathfrak{m}$-enabled at C, to a configuration $C' = (w'_1, \ldots w'_m)$ such that, for each compartment i of μ:

$$w'_i = w_i \backslash lhs_i^{\mathbf{r}} \cup \{a \in V \mid a \in rhs_i^{\mathbf{r}} \vee a_{in_i} \in rhs_{parent(i)}^{\mathbf{r}} \vee \exists (i,j) \in \mu : a_{out} \in rhs_j^{\mathbf{r}}\} \ .$$

(It is assumed that $rhs_{parent(i)}^{\mathbf{r}} = \varnothing$ if i is the root of μ.)
We denote this by $C \stackrel{\mathbf{r}}{\longrightarrow}_{\mathfrak{m}} C'$. An $\mathfrak{m}$-*computation* is then defined as a (finite or infinite) sequence of consecutive $\mathfrak{m}$-evolutions starting from C_0.

The difference between the 'qualitative' and the 'quantitative' interpretation of the evolution rules is twofold. First, there may be two enabled evolution rules in a compartment with a common object in their left hand sides while there is only a single representant of that object in the current state in the compartment. In the current qualitative set-up, the two rules can be executed together. That is, objects are characterised by their presence rather then their quantity. Second, if two simultaneously executed rules produce the same object in the same compartment, instead of adding two instances of this object, only one is added (so that we never have more than a single representant of an object in any given compartment). As a consequence, there is no need to use multisets of objects present in any single compartment to represent the current state, and there is no need to use vectors of multisets of rules in set membrane systems. In either case, using sets is fully sufficient. One may observe that with this view of state representation and system execution, *max*-evolution is *deterministic* in set membrane systems. Other kinds of evolutions can be non-deterministic in the sense that there may be different vector set-rules executed at a given configuration C. Figure 3 shows a two-stage lmax-evolution for the example shown in Figure 2.

3 SET-nets with localities

We now introduce *basic set-nets with localities* (or BSL-nets), the new class of Petri nets that provides in a natural way a model for the behaviour of basic set

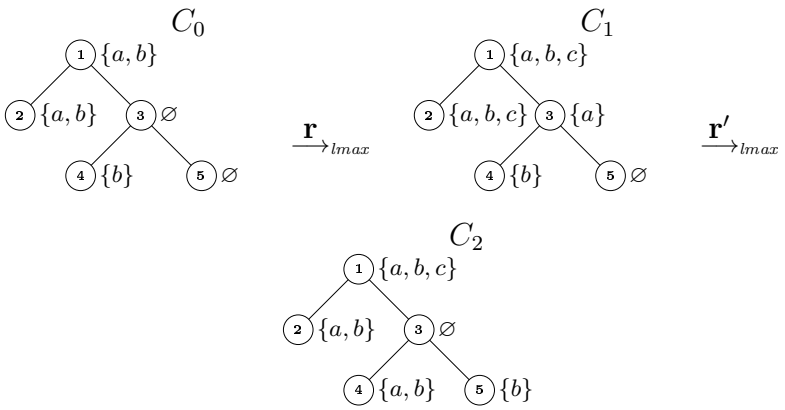


Fig. 3. An lmax-computation for the running example with the vector set-rules defined in the following way: $\mathbf{r} = \langle\{r_{11}, r_{12}, r_{13}\}, \varnothing, \varnothing, \varnothing, \varnothing\rangle$ and $\mathbf{r}' = \langle\varnothing, \{r_{21}, r_{22}\}, \{r_{31}\}, \varnothing, \varnothing\rangle$.

membrane systems. The BSL-net model is derived from the recently introduced SET-nets [11] developed as a model for reaction systems [2–4]. In addition, similar to the Petri net model corresponding to quantitative membrane systems, transitions in BSL-nets transitions belong to localities which influences the ensuing execution semantics.

A BSL-*net* is a tuple $N = (P, T, F, \ell, M_0)$ such that P and T are finite disjoint sets of respectively places and transitions, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, $\ell : T \to \mathbb{N}$ is the *locality mapping*; in general, any set of places is a *marking* and $M_0 \subseteq P$ is the *initial* marking of N.
We use the standard dot-notation: ${}^\bullet x = \{y \mid (y, x) \in F\}$ for the inputs, and $x^\bullet = \{y \mid (x, y) \in F\}$ for the outputs, of a given place or transition x. We lift this notation in the usual way to sets U of transitions, i.e., ${}^\bullet U = \bigcup_{t \in U} {}^\bullet t$ and $U^\bullet = \bigcup_{t \in U} t^\bullet$.

In diagrams, like that in Figure 4, places are drawn as circles, and transitions as boxes. If $(x, y) \in F$ then (x, y) is an *arc* leading from x to y. A marking M is represented by drawing in each place $p \in M$ a token (a small black dot). Boxes representing transitions belonging to the same localities are displayed on a grey background of the same shade. Note that the locality mapping ℓ partitions the transition set by associating with each transition a locality, in this case a compartment.

As in SET- nets, there is no concept of token counting in BSL-nets. In this sense they resemble elementary net systems (EN-systems) [16], a fundamental model to study basic features of concurrent systems. However, the execution semantics is strikingly different. When a place of a SET-net is marked, this indicates nothing but non-emptiness or presence of a resource without any quantification. Consequently, this place can be seen as providing input to any number of transitions at

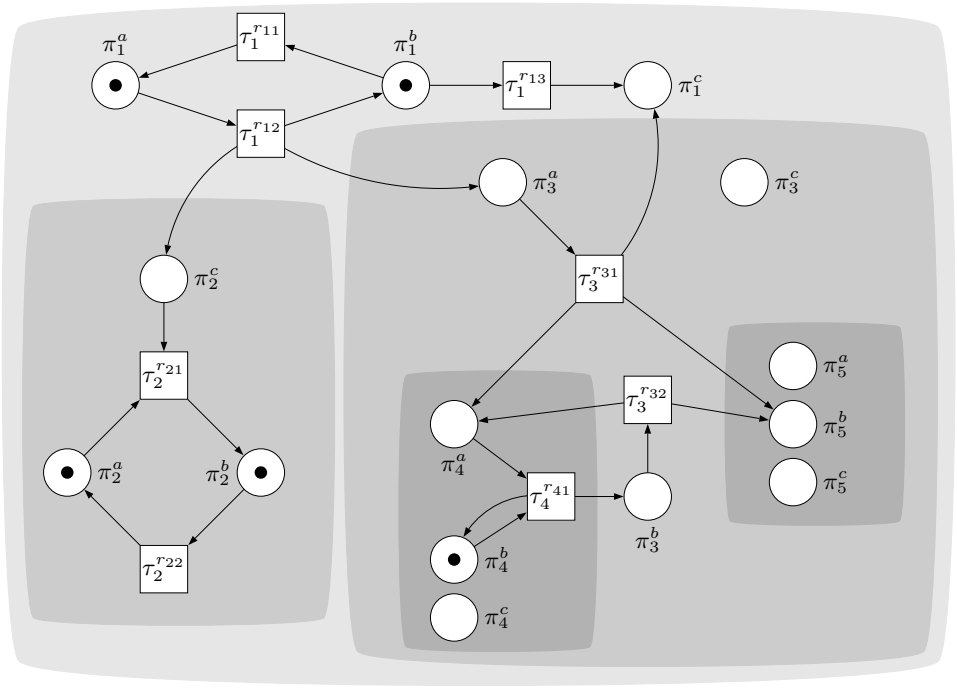


Fig. 4. BSL-net corresponding to the running example.

the same time (and as a result, there are no conflicts between transitions sharing an input place). Firing a transition empties all its input places and marks each of its output places, again without any further logical interpretation or quantification. Hence, again in contrast with EN-systems, a transition can fire when it has a non-empty output place and it can also fire simultaneously with transitions with which it shares an output place.

In a BSL-net N as above, transition $t \in T$ can occur (is enabled) at a marking M if ${}^\bullet t \subseteq M$. If t is enabled at M and is executed this leads to a marking M' given by $M' = (M \setminus {}^\bullet t) \cup t^\bullet$. Moreover, similar to the reactions in vector set-rules, transitions may occur simultaneously as *steps*. It should be noted here, that now — in contrast to the steps in PT-systems or the vector rules in quantitative membrane systems — multiple occurrences of the same transition in a step are not allowed, i.e., steps are sets. In fact, since SET-nets are non-counting, executing multiple copies of the same transition has exactly the same effect as executing a single it just once.
As for basic set membrane systems, we distinguish four modes of execution.

A step $U \subseteq T$ is

– *free-enabled* at a marking M if each transition in U is enabled.

Moreover, a free-enabled step U is:

- *min-enabled* if $|U| = 1$;
- *max-enabled* if U comprises all transitions enabled at M; and
- *lmax-enabled* if U comprises all transitions t enabled at M with $\ell(t) \in \ell(U)$.

Note that a step is enabled at a marking M if all input places of its transitions are marked. For the BSL-net in Figure 4 and its initial marking, we have that:

- $\{\tau_2^{r_{21}}\}$ is not free-enabled;
- $\{\tau_1^{r_{11}}\}$ is min-enabled but not lmax-enabled;
- $\{\tau_1^{r_{11}}, \tau_1^{r_{12}}, \tau_1^{r_{13}}\}$ is lmax-enabled but not max-enabled; and
- $\{\tau_1^{r_{11}}, \tau_1^{r_{12}}, \tau_1^{r_{13}}, \tau_2^{r_{22}}\}$ is max-enabled.

A step U which is $\mathfrak{m}$-enabled at a marking M can be $\mathfrak{m}$-*executed* leading to another marking M' given by $M' = (M \backslash {}^\bullet U) \cup U^\bullet$. We denote this by $M\ [U\rangle_{\mathfrak{m}}\ M'$. An $\mathfrak{m}$-*computation* of N is then a (finite or infinite) sequence of $\mathfrak{m}$-executions starting from M_0. A possible two-stage lmax-computation for the BSL-net of Figure 4 is:

$$M_0\ [\{\tau_1^{r_{11}}, \tau_1^{r_{12}}, \tau_1^{r_{13}}\}\rangle_{lmax}\ M'\ [\{\tau_2^{r_{21}}, \tau_2^{r_{22}}, \tau_3^{r_{31}}\}\rangle_{lmax}\ M\ , \qquad (\dagger)$$

where $M' = \{\pi_1^a, \pi_1^b, \pi_1^c, \pi_2^a, \pi_2^b, \pi_2^c, \pi_3^a, \pi_4^b\}$ and $M = \{\pi_1^a, \pi_1^b, \pi_1^c, \pi_2^a, \pi_2^b, \pi_4^a, \pi_4^b, \pi_5^b\}$.

4 From basic set membrane systems to BSL-nets

To model a basic membrane system as a BSL-net, we construct a separate place π_j^a, for each object a and membrane $j \in \mu$. Moreover, for each evolution rule r associated with a membrane i, we introduce a transition τ_i^r with locality i. If the transformation described by an evolution rule r of compartment i consumes a, then we introduce an arc from place π_i^a to transition τ_i^r, and similarly for objects being produced. Finally, we put a token into place π_j^a whenever compartment j contains initially object a. Formally, we proceed as follows.

Given a basic set membrane system $\Sigma = (V, \mu, w_1^0, \ldots, w_m^0, R_1, \ldots, R_m)$ over the membrane structure μ, the BSL-net *corresponding to* Σ is $N_\Sigma = (P, T, F, \ell, M_0)$, where the places, transitions and the initial marking are respectively given by:

$$\begin{aligned} P &= \{\pi_i^a \mid i \leq m \wedge a \in V\} \\ T &= \{\tau_i^r \mid i \leq m \wedge r \in R_i\} \\ M_0 &= \{\pi_i^a \mid i \leq m \wedge a \in w_i^0\}\ , \end{aligned}$$

and, for every transition $\tau = \tau_i^r$, we have $\ell(\tau) = i$ as well as:

$$\begin{aligned} {}^\bullet\tau &= \{\pi_i^a \mid a \in lhs^r\} \\ \tau^\bullet &= \{\pi_i^a \mid a \in rhs^r\} \ \cup\ \{\pi_j^a \mid a_{in_j} \in rhs^r\} \ \cup\ \{\pi_{parent(j)}^a \mid a_{out} \in rhs^r\}\ . \end{aligned}$$

Figure 4 shows the translation for the running example.

The tight correspondence between the membrane system Σ and the BSL-net N_Σ is captured by a translation from configurations of Σ to markings of N_Σ, based on the correspondence of object locations and places as well as the correspondence of vector set-rules and steps. More precisely, the marking $\nu(C)$ *corresponding* to a configuration $C = (w_1, \ldots, w_m)$ of Σ is defined by $\nu(C) = \{\pi_i^a \mid i \leq m \wedge a \in w_i\}$, and the step $\rho(\mathbf{r})$ *corresponding* to a vector set-rule $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$ of Σ by $\rho(\mathbf{r}) = \{\tau_i^r \mid r \in \mathbf{r}_i\}$. For example, if we take the lmax-computation of the running example given in Figure 3, and the lmax-computations of the corresponding BSL-net given in (†), then we have $\rho(\mathbf{r}') = \{\tau_2^{r_{21}}, \tau_2^{r_{22}}, \tau_3^{r_{31}}\}$ and $\nu(C_2) = M$. It follows directly from the definitions that ν and ρ are bijections with the initial configuration of Σ corresponding to the initial marking of N_Σ.

Proposition 1. *The two mappings, ν and ρ, are two bijections such that, for every marking M of N_Σ:*

$$\nu^{-1}(M) = (\{a \mid \pi_1^a \in M\}, \ldots, \{a \mid \pi_m^a \in M\}),$$

and, for every step U of N_Σ, we have $\rho^{-1}(U) = \langle \{r \mid \tau_1^r \in U\}, \ldots, \{r \mid \tau_m^r \in U\} \rangle$.

Proposition 2. $\nu(C_0) = M_0$.

For a translation from one dynamic system to another to be useful, it is essential to ensure that the latter provides a faithful representation of the behaviour of the former. Here, it is possible to establish the desired relationship between the operation of set membrane systems and BSL-nets at the system level. The fundamental link between the dynamics of a set membrane system and that of its corresponding BSL-net is formulated next.

Theorem 1. *Given a set membrane system Σ and the corresponding BSL-net N_Σ, we have that:*

$$C \xrightarrow{\mathbf{r}}_{\mathfrak{m}} C' \;\; \textit{in} \;\; \Sigma \quad \textit{if and only if} \quad \nu(C)\,[\rho(\mathbf{r})\rangle_{\mathfrak{m}}\,\nu(C') \;\; \textit{in} \;\; N_\Sigma\,,$$

for each mode of execution $\mathfrak{m}$.

Proof. Below $C = (w_1, \ldots, w_m)$, $C' = (w_1', \ldots, w_m')$ and $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$. We will first show that

$$\mathbf{r} \text{ is min-enabled at } C \quad \text{iff} \quad \rho(\mathbf{r}) \text{ is min-enabled at } \nu(C)\,.$$

Indeed, in such a case, there are an $i \leq m$ and an $r \in R_i$ such that $\mathbf{r}_i = \{r\}$ and $\mathbf{r}_j = \varnothing$, for all $j \neq i$. Hence $\rho(\mathbf{r}) = \{\tau\}$ where $\tau = \tau_i^r$. Let $lhs^r = \{a^1, \ldots, a^k\}$ which means that ${}^\bullet\tau = \{\pi_i^{a^1}, \ldots, \pi_i^{a^k}\}$. We than have:

$$\begin{array}{c}\mathbf{r} \text{ is min-enabled at } C \quad \text{iff} \quad \{a^1, \ldots, a^k\} \subseteq w_i \quad \text{iff} \quad \{\pi_i^{a^1}, \ldots, \pi_i^{a^k}\} \subseteq \nu(C) \\ \text{iff} \quad {}^\bullet\tau \subseteq \nu(C) \quad \text{iff} \quad \rho(\mathbf{r}) \text{ is min-enabled at } \nu(C)\,.\end{array}$$

In view of what we have just established, and the fact that the enabledness of a set of evolution rules (transitions) is equivalent to the enabledness of individual evolution rules (transitions), we immediately obtain that:

$$\mathbf{r} \text{ is free-enabled at } C \quad \text{iff} \quad \rho(\mathbf{r}) \text{ is free-enabled at } \nu(C) \ .$$

Moreover, given that the locality of the transition τ_i^r corresponding to an evolution rule $r \in R_i$ is i, it follows that, for every execution mode $\mathfrak{m}$:

$$\mathbf{r} \text{ is } \mathfrak{m}\text{-enabled at } C \quad \text{iff} \quad \rho(\mathbf{r}) \text{ is } \mathfrak{m}\text{-enabled at } \nu(C) \ .$$

All what remains now to be shown is that the executions of $\mathbf{r}$ and $\rho(\mathbf{r})$ lead to equivalent results. This, however, is clearly the case given the way the results of the executions of vector set-rules and steps of transitions are defined as well as the equivalence stemming from the executions of a single evolution rule and the corresponding transition.

To demonstrate the latter point, let us consider an evolution rule $r \in R_i$ and the corresponding transition $\tau = \tau_1^r$. Moreover, let $lhs^r = \{a^1, \ldots, a^k\}$ and:

$$rhs^r = \{b^1, \ldots, b^n\} \cup \{c^1_{out}, \ldots, c^s_{out}\} \cup \{d^{11}_{in_{j_1}}, \ldots, d^{1q_1}_{in_{j_1}}, \ldots, d^{p1}_{in_{j_p}}, \ldots, d^{pq_{j_p}}_{in_{j_p}}\} \ ,$$

where $j_z \neq j_x$ for $z \neq x$. Then, by the definition of N_Σ, we have:

$$\begin{aligned} {}^\bullet\tau &= \{\pi_i^{a^1}, \ldots, \pi_i^{a^k}\} \\ \tau^\bullet &= \{\pi_i^{b^1}, \ldots, \pi_i^{b^n}\} \ \cup \ \{\pi^{c^1}_{parent(i)}, \ldots, \pi^{c^s}_{parent(i)}\} \\ &\qquad \cup \ \{\pi^{d^{11}}_{j_1}, \ldots, \pi^{d^{1q_{j_1}}}_{j_1}, \ldots, \pi^{d^{p1}}_{j_p}, \ldots, \pi^{d^{pq_{j_p}}}_{j_p}\} \ . \end{aligned}$$

We then observe that executing $\mathbf{r}$ such that $\mathbf{r}_i = \{r\}$ and $\mathbf{r}_j = \varnothing$, for all $j \neq i$, leads to a configuration C' such that, for all $x \leq m$:

$$w'_x = \begin{cases} (w_x \setminus \{a^1, \ldots, a^k\}) \cup \{b^1, \ldots, b^n\} & \text{if } x = i \\ w_x \cup \{c^1, \ldots, c^s\} & \text{if } x = parent(i) \\ w_x \cup \{d^{11}, \ldots, d^{1q_1}\} & \text{if } x = j_1 \\ \ldots & \ldots \\ w_x \cup \{d^{p1}, \ldots, d^{pq_p}\} & \text{if } x = j_p \\ w_x & \text{otherwise} \ . \end{cases}$$

It is then not difficult to check that:

$$\begin{aligned} \nu(C') = \nu(C) \setminus \ &\{\pi_i^{a^1}, \ldots, \pi_i^{a^k}, \pi_i^{b^1}, \ldots, \pi_i^{b^n}, \pi^{c^1}_{parent(i)}, \ldots, \pi^{c^s}_{parent(i)}\} \\ &\cup \{\pi^{d^{11}}_{j_1}, \ldots, \pi^{d^{1q_{j_1}}}_{j_1}, \ldots, \pi^{d^{p1}}_{j_p}, \ldots, \pi^{d^{pq_{j_p}}}_{j_p}\} \ . \end{aligned}$$

which is exactly $(\nu(C) \setminus {}^\bullet\tau) \cup \tau^\bullet$, as required by the equivalence result.

Together with Propositions 1 and 2, this means that the (finite and infinite) $\mathfrak{m}$-computations of the basic set reaction system Σ coincide with the (finite and infinite) $\mathfrak{m}$-computations of the corresponding BSL-net N_Σ.

5 Set membrane systems with promoters and inhibitors

Basic (quantitative) membrane systems have over the past decade been extended in several different directions, motivated either by their potential applications, or by their computational properties. For some of these extensions, like catalysts and symport/antiport rules, there exist straightforward translation to Petri nets (see, for example, [5]). For others, like i/o communication and rule creation/consumption, the correspondence between evolution rules and Petri net transitions is more involved, and the resulting nets are additionally equipped with inhibitor and/or activator arcs (see, e.g., [8]).

Given the nature of many biochemical reactions, we feel that presumably a key extension is one allowing evolution rules to be triggered or blocked by the presence of certain objects. In fact, this is exactly the view followed in the reaction system model which inspired the work presented in this paper. To capture such an extension in set membrane systems, we consider evolution rules r of the form:

$$lhs^r \to rhs^r|_{pro^r\,,\,inh^r}$$

where pro^r and inh^r are sets of objects specifying respectively the *promoters* and *inhibitors* or r. This definition is derived from [13] where (multisets of) promoters and inhibitors were considered in the context of (quantitative) membrane systems. The intuition behind pro^r and inh^r is that they only test for the presence and absence, respectively, of certain objects inside a compartment.

In order for r to occur, each object in pro^r must be present in its associated compartment, and each object in inh^r must be absent. In the formalisation of the extended evolution rules we retain the definitions and notations introduced for the set basic membrane systems, except for the notion of a free-enabled (and its derivations of min-enabled, max-enabled and lmax-enabled) vector set-rule $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$. This is strengthened by additionally requiring that, for each membrane $i \in \mu$ and evolution rule $r \in \mathbf{r}_i$, we have $pro_i^r \subseteq w_i$ and $inh^r \cap w_i = \varnothing$. The resulting Σ is called a *set membrane reaction system (with promoters and inhibitors)*. We then extend the BSL-net model to provide a matching class of nets.

ESL-nets

An *extended set-net with localities* (or ESL-net) is a tuple

$$N = (P, T, F, Inh, Act, \ell, M_0)$$

such that (P, T, F, ℓ, M_0) is a BSL-net and the two new components, $Inh \subseteq P \times T$ and $Act \subseteq P \times T$, are its sets of *inhibitor* and *activator* arcs. We also denote ${}^\circ U = \{p \mid \exists t \in U : (p,t) \in Inh\}$ and ${}^\blacklozenge U = \{p \mid \exists t \in U : (p,t) \in Act\}$, for every set of transitions U. The definitions and notations concerning the marking change in N are the same as for the underlying BSL-net (P, T, F, ℓ, M_0) with one

exception, namely a set of transitions U is free-enabled at a marking M if we have:

$$^\bullet U \cup {}^\blacklozenge U \subseteq M \quad \text{and} \quad {}^\circ U \cap M = \varnothing \, .$$

Thus, each place connected by an activator arc to a transition in U should carry a token, while each place connected by an inhibitor arc to a transition in U should be empty. The notions of (finite or infinite) $\mathfrak{m}$-computations of N for the four distinguished execution modes $\mathfrak{m}$ are then defined as before.

From set membrane systems to ESL-nets

The translation from set membrane systems with promoters and inhibitors to ESL-nets proceeds as in the case of the basic set membrane system. The only additional feature is that for each transition τ_i^r and place π_i^a, we introduce an inhibitor arc (π_i^a, τ_i^r) whenever $a \in inh^r$, and we introduce an activator arc (π_i^a, τ_i^r) whenever $a \in pro^r$. It then turns out that the properties of the extended translation are very similar to those obtained in the basic case; in particular, we obtain the following.

Theorem 2. *Given a set membrane system with promoters and inhibitors Σ and the corresponding* ESL*-net N_Σ, we have that:*

$$C \xrightarrow{\mathbf{r}}_{\mathfrak{m}} C' \;\; \textit{in} \;\; \Sigma \quad \textit{if and only if} \quad \nu(C)\,[\rho(\mathbf{r})\rangle_{\mathfrak{m}}\,\nu(C') \;\; \textit{in} \;\; N_\Sigma \, ,$$

for each mode of execution $\mathfrak{m}$.

Proof. Similar to the proof of Theorem 1. The impact of promoters/inhibitors and activator/inhibitor arcs on the enabledness of an evolution rule and the corresponding transition is equivalent. Moreover, the resulting configuration and marking do not depend on promoters/inhibitors nor activator/inhibitor arcs.

6 Concluding remarks

As already in the introduction, qualitative membrane systems were independently introduced in [1] with the aim of characterising their language theoretic properties. The present paper looked at such a model from a totally different perspective, focussing on aspects relating to different semantical interpretations, and the relationship to Petri nets.

Moving from quantitative to qualitative membrane systems is an abstraction which may lead to a more tractable approach when it comes to answering vital questions concerning the evolution of systems. However, to take advantage of this fact, the existing concrete analysis tools developed for the classical, quantitative, Petri net models need to be adapted for set-nets.

For one thing, the process concept underlying the causality semantics of standard Petri net models (see, e.g., [7]) has to be reconsidered. As can be seen from examples in [11], the cause and effect relation in set-nets (and hence

membrane systems with qualitative evolution rules) will have to be interpreted in a completely new fashion.

In [9], we have already made preliminary investigation into the *synthesis problem* which aims at an automatic construction of set-nets exhibiting a behaviour given in terms of a transition system. For set membrane systems this should contribute to insight in which evolution rules lead to certain observed behaviour.

Finally, by bringing qualitative (set rather than multiset) aspects to membrane systems, also interesting questions relating to expressive (generative) power emerge. For every mode, one can consider the possible evolutions of a system of a set membrane system (i.e., the computations of BSL-nets) as a language. These languages are regular subset languages. The study of subset languages of Petri nets was initiated in [17, 18] but still for the standard (quantitative) interpretation. There are a number of interesting theoretical questions and topics for the regular subset languages generated by BSL/ESL-nets under the four execution modes as well all regular subset languages. For example, one can consider: inclusion hierarchies; closure properties; and the complexity of equivalence/inclusion checking. Another group of problems here would be motivated by the target application area, i.e., biochemistry. For example, one can investigate: oscillatory behaviour (is it possible to have cycles from some point with at least/at most/specific evolution rules only); or vitality of the system (possible deadlock or partial death, i.e., some rules that can no longer be executed) or other state-related properties, like whether it would be possible for two objects to appear in a given compartment at some point together.

Acknowledgement We would like to thank Grzegorz Rozenberg for explaining to us the ideas underlying reaction systems.

References

1. Alhazov, A.: P Systems Without Multiplicities of Symbol-objects. Information Processing Letters **100** (2006) 124–129
2. Ehrenfeucht, A., Main, M., Rozenberg, G.: Combinatorics of Life and Death for Reaction Systems. International Journal of Foundations of Computer Science **22** (2009) 345–356
3. Ehrenfeucht, A., Rozenberg, G.: Reaction Systems. Fundamenta Informaticae **76** (2006) 1–18
4. Ehrenfeucht, A., Rozenberg, G.: Events and Modules in Reaction Systems. Theoretical Computer Science **376** (2007) 3–16
5. Ibarra, O.H., Dang, Z., Egecioglu, O.: Catalytic P Systems, Semilinear Sets, and Vector Addition Systems. Theoretical Computer Science **312** (2004) 379–399
6. Ibarra, O.H., Ye, H.C., Dang, Z.: The Power of Maximal Parallelism in P Systems. Lecture Notes in Computer Science **3340** (2004)
7. Kleijn, H.C.M., Koutny, M.: Process Semantics of General Inhibitor Nets. Information and Computation **190** (2004) 18–69
8. Kleijn, J., Koutny, M.: Processes of Membrane systems with Promoters and Inhibitors. Theoretical Computer Science **404** (2008) 112–126

9. Kleijn, H.C.M., Koutny, M., Pietkiewicz-Koutny, M., Rozenberg, G.: Region Based Set-net Synthesis. Technical Report (2011)
10. Kleijn, H.C.M., Koutny, M., Rozenberg, G.: Towards a Petri Net Semantics for Membrane Systems. In: Freund, R., Paun, G., Rozenberg, G., Salomaa, A. (eds.): WMC 2005. Lecture Notes in Computer Science **3850**. Springer-Verlag, Berlin Heidelberg New York (2006) 292–309
11. Kleijn, H.C.M., Koutny, M., Rozenberg, G.: Modelling Reaction Systems with Petri Nets. Technical Report (2011)
12. Păun, G.: Computing with Membranes. J. Comput. Syst. Sci. **61** (2000) 108–143
13. Păun, G.: Membrane Computing, An Introduction. Springer-Verlag, Berlin Heidelberg New York (2002)
14. Păun, G., Rozenberg, G.: A Guide to Membrane Computing. Theoretical Computer Science **287** (2002) 73–100
15. Păun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press (2009)
16. Rozenberg, G., Engelfriet, J.: Elementary Net Systems. Lectures on Petri Nets I: Basic Models, W.Reisig and G.Rozenberg (eds.), Lecture Notes in Computer Science **1491**, Springer-Verlag, Berlin Heidelberg New York (2006) 173–187
17. Rozenberg, G., Verraedt, R.: Subset Languages of Petri Nets Part I: The Relationship to String Languages and Normal Forms. Theoretical Computer Science **26** (1983) 301–326
18. Rozenberg, G., Verraedt, R.: Subset Languages of Petri Nets Part II: Closure Properties. Theoretical Computer Science **27** (1983) 85–108
19. Membrane systems web page (2011) `http://ppage.psystems.eu/`