Scientific Annals of Computer Science vol. 21, 2011, pp. 199–225

Petri Nets for Biologically Motivated Computing

Jetty KLEIJN[1], Maciej KOUTNY[2], Grzegorz ROZENBERG[3]

Abstract

Petri nets are a general and well-established model of concurrent and distributed computation and behaviour, including that taking place in biological systems. In this survey paper, we are concerned with intrinsic relationships between Petri nets and two formal models inspired by aspects of the functioning of the living cell: membrane systems and reaction systems. In particular, we are interested in the benefits that can result from establishing strong semantical links between Petri nets and membrane systems and reaction systems. We first discuss Petri nets with localities reflecting the compartmentalisation modelled in membrane systems. Then special attention is given to set-nets, a new Petri net model for reaction systems and their qualitative approach to the investigation of the processes carried out by biochemical reactions taking place in the living cell.

Keywords: Petri net, biomodelling, membrane system, reaction system, locality, GALS, qualitative modelling, set-net, set membrane system.

1 Introduction

Petri nets (see e.g., [39]) are a general formal model for concurrent and distributed computation. Over the years, an impressive variety of Petri net models suited for many different applications have been developed together with supporting theories and tools. A relatively new and ever more important field of application is biology. Thanks to its distributed character, the

[1]LIACS, Leiden University, 2300 RA, The Netherlands. Email: `kleijn@liacs.nl`

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

[3]LIACS, Leiden University, 2300 RA, The Netherlands, and Department of Computer Science, University of Colorado at Boulder, U.S.A. Email: `rozenber@liacs.nl`

Petri net approach appears to be particularly well-suited to provide computational and operational foundations for problems and issues arising in biology; see for example, [28], for a recent comprehensive overview of applications of Petri nets in systems biology. On the other hand, to understand or make use of specific aspects of biological processes, new formal models have been proposed. Membrane systems and reaction systems are two examples of such models which are both abstractions of the functioning of the living cell.

In this paper, we are concerned with intrinsic similarities and differences between Petri nets on the one hand, and membrane systems and reaction systems on the other hand. In particular, we are interested in establishing strong semantical links between these two models and Petri nets, and the possible mutual benefits that may result. Different enhancements of the Petri net model are considered for the faithful modelling of the dynamics of the biological phenomena represented by membrane systems and reaction systems. It is our aim to demonstrate the fruitful two-way interaction between Petri nets and the other two models. We will be interested in Petri net semantics which open the way to importing Petri net methodologies and tools to the two biologically motivated computational models. Such semantics should be faithful, so rather than giving a Petri net interpretation, we adapt and incorporate new concepts into the Petri net framework, while retaining the underlying Petri net philosophy. In particular, we will discuss an extension of the standard PT-nets with a concept of locality, and much emphasis will be given to the completely new class of Petri nets called SET-nets. The latter are suited for qualitative rather than quantitative modelling which plays an important role in rendering of the biochemical processes that take place in living cells.

Petri nets are a graphical modelling language with strong mathematical, algorithmic and tool support for the specification and analysis of distributed systems. Many different classes of Petri nets have been developed since their first appearance in [37]. Their main common underlying philosophy is that states are distributed and actions have a local cause and effect (on the adjacent components of the net); for more discussions on this see [7]. The most typical Petri nets are without doubt the Place/Transition nets (or PT-nets) [8]. They are based on the production and consumption of resources by actions taking place in the system. Hence the resulting computational processes are essentially multiset based. Another well known, more fundamental, Petri net class are the Elementary Net systems (or EN-systems) [8].

Their dynamics is based on holding or not holding of local conditions rather than being resource based.

Like PT-nets, membrane systems ([33, 34, 35, 36]) are essentially multiset rewriting systems. As a computational model they are inspired by the way chemical reactions take place in cells which are divided by membranes into compartments. The reactions are abstracted to rules that specify which and how many molecules can be produced from given molecules of a certain kind and quantity. The dynamic aspects of the membrane system model including potential behaviour (computations), derive from such evolution rules. To capture the compartmentalisation of membrane systems, PT-nets are extended with transition localities. This makes it possible to have locally synchronised executions, but it requires an extension of the causality semantics of PT-nets.

Reaction systems [3, 10, 11, 12] are also a model for the investigation of processes carried out by biochemical reactions in living cells. The model is meant to contribute to the understanding of the interactions between such reactions. This time, however, biochemical reactions are based on qualitative rather than quantitative presence of resources. Hence, in order to obtain a faithful Petri net representation of reaction systems, it is necessary to re-evaluate the existing Petri net modelling approaches. We therefore introduce a new class of Petri nets, called SET-nets, that supports set-based (boolean) operations on tokens rather than the standard Petri net multiset-based token manipulation.

Finally, we bring together the qualitative approach of reaction systems and the compartmentalisation of membrane systems as the multiset approach of membrane systems is not always realistic from the point of view of explicitly counting huge number of molecules and reactions. Moreover, the resulting infinite state space makes it impractical or impossible to apply formal verification techniques. Therefore, we propose to consider set membrane systems, that is membrane systems with qualitative evolution rules. This is especially attractive as localities and SET-nets can be combined to yield a satisfactory Petri net semantics for set membrane systems.

In this survey paper, we mainly describe approaches and give the essence of key results. More details on Petri nets and membrane systems can be found in [23]. Set membrane systems were introduced in [24], and SET-nets were first presented at the BioPPN Workshop held in Newcastle upon Tyne in June 2011 (see [27] for the informal workshop version).

2 Preliminaries

Multisets A multiset over a set X is a function $\mu : X \to \mathbb{N} = \{0, 1, 2, \ldots\}$. (In this paper we only consider the case that X is finite.) Multiset μ is said to be empty if there are no x such that $x \in \mu$ by which we mean that $x \in X$ and $\mu(x) \geq 1$. The empty multiset is denoted by $\varnothing$.

A multiset may be represented by listing its elements with repetitions, e.g., $\mu = \{y, y, z\}$ is such that $\mu(y) = 2$, $\mu(z) = 1$, and $\mu(x) = 0$ otherwise. We treat sets as multisets without repetitions.

For two multisets μ and μ' over X, the sum $\mu + \mu'$ is the multiset given by $(\mu + \mu')(x) = \mu(x) + \mu'(x)$ for all $x \in X$, and if $k \in \mathbb{N}$ then $k \cdot \mu$ is the multiset given by $(k \cdot \mu)(x) = k \cdot \mu(x)$ for all $x \in X$. The difference $\mu - \mu'$ is given by $(\mu - \mu')(x) = \max\{\mu(x) - \mu'(x), 0\}$ for all $x \in X$. We denote $\mu \leq \mu'$ whenever $\mu(x) \leq \mu'(x)$ for all $x \in X$, and $\mu < \mu'$ whenever $\mu \leq \mu'$ and $\mu \neq \mu'$.

Petri Nets A *Place/Transition net* (or PT-net) is defined as a tuple

$$PT = (Pl, Tr, W, M_0) \ ,$$

where: Pl and Tr are finite disjoint sets of respectively *places* and *transitions*; $W : (Tr \times Pl) \cup (Pl \times Tr) \to \mathbb{N}$ is the *arc weight function*; and $M_0 : Pl \to \mathbb{N}$ is the *initial marking* (in general, any multiset of places is a marking).

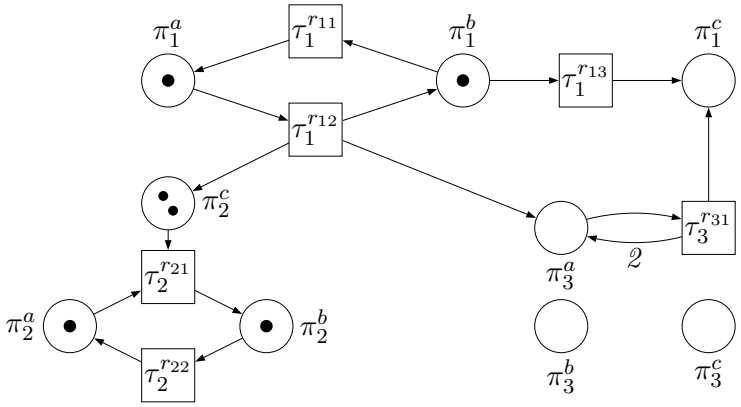

Figure 1: PT-net.

In diagrams, like that in Figure 1, places are drawn as circles, and transitions as boxes. If $W(x, y) \geq 1$, then (x, y) is an *arc* leading from x to

y. An arc is annotated with its weight if the latter is greater than one. A marking M is represented by drawing in each place p exactly $M(p)$ tokens (small black dots).

A *step* U of PT is a multiset of transitions. Its *pre-multiset* and *post-multiset* of places, ${}^\bullet U$ and $U^\bullet$, are respectively given by

$$ {}^\bullet U(p) = \sum_{t \in U} U(t) \cdot W(p,t) \quad \text{and} \quad U^\bullet(p) = \sum_{t \in U} U(t) \cdot W(t,p) \ , $$

for each place p. For the PT-net in Figure 1 we have:

$${}^\bullet\{\tau_1^{r_{11}}, \tau_1^{r_{11}}, \tau_3^{r_{31}}\} = \{\pi_1^b, \pi_1^b, \pi_3^a\} \text{ and } \{\tau_1^{r_{11}}, \tau_1^{r_{11}}, \tau_3^{r_{31}}\}^\bullet = \{\pi_1^a, \pi_1^a, \pi_1^c, \pi_3^a, \pi_3^a\}.$$

We can distinguish three modes of execution for PT-nets (from sequential to fully synchronous). To start with, U is *free-enabled* at a marking M if ${}^\bullet U \leq M$. A free-enabled U is then: *min-enabled* if $|U| = 1$; and *max-enabled* if U cannot be extended by a transition to yield a step which is free-enabled at M. For the PT-net in Figure 1 we have that, at the initial marking M_0, the step $\{\tau_1^{r_{12}}, \tau_2^{r_{21}}\}$ is free-enabled, $\{\tau_1^{r_{11}}\}$ is min-enabled, and $\{\tau_1^{r_{11}}, \tau_1^{r_{12}}, \tau_2^{r_{21}}, \tau_2^{r_{22}}\}$ is max-enabled.
That is, U is free-enabled at M if in each place there are enough tokens for the specified multiple occurrence of each of its transitions (note that each transition t needs to consume from each place p exactly $W(p,t)$ tokens which cannot be shared with any other transition). Interleaving (min-enabledness) allows only one transition to be executed at a time. Maximal concurrency (max-enabledness) means that extending U would demand more tokens than M supplies.

For each mode of execution $\mathfrak{m} \in \{\mathit{free}, \mathit{min}, \mathit{max}\}$, a step U which is $\mathfrak{m}$-enabled at a marking M can be $\mathfrak{m}$-*executed* leading to the marking M' given by

$$ M' = M - {}^\bullet U + U^\bullet \ . $$

We denote this by $M[U\rangle_{\mathfrak{m}} M'$. Moreover, an $\mathfrak{m}$-*step sequence* is a finite or infinite sequence of $\mathfrak{m}$-executions starting from the initial marking. For the PT-net in Figure 1 we have

$$ M_0[\{\tau_1^{r_{12}}, \tau_2^{r_{21}}\}\rangle_{\mathit{free}} \{\pi_1^b, \pi_1^b, \pi_2^b, \pi_2^b, \pi_2^c, \pi_2^c, \pi_3^a\} \ . $$

Petri Nets with Inhibitor and Activator Arcs A PT-net can be equipped with two other kinds of arcs which *test* for the presence or absence of tokens in places. More precisely,

$$ \mathit{Inh} \subseteq \mathit{Pl} \times \mathit{Tr} \quad \text{and} \quad \mathit{Act} \subseteq \mathit{Pl} \times \mathit{Tr} $$

are respectively the sets of *inhibitor* and *activator* arcs. In diagrams, an inhibitor arc ends with a small open circle, while an activator arc ends with a small black circle. The role of both kinds of test arcs is to constrain the enabling of a step U by stipulating that it is *free-enabled* at a marking M if ${}^\bullet U \leq M$ as well as

- $p \notin M$ whenever there is $t \in U$ such that $(p,t) \in \mathit{Inh}$
- $p \in M$ whenever there is $t \in U$ such that $(p,t) \in \mathit{Act}$.

All the remaining notions are the same as for PT-nets.

Petri Nets with Localities A *Place/Transition net with localities* (or PTL-net) is defined as a tuple $PTL = (Pl, Tr, W, \ell, M_0)$ such that (Pl, Tr, W, M_0) is a PT-net and $\ell : Tr \to \mathbb{N}$ is a *locality mapping*. In diagrams, such as that in Figure 4, boxes representing transitions belonging to the same localities are displayed on a grey background of the same shade.

Localities can be used to define one more kind of enabling for steps of transitions. A *step* U of PTL is *lmax-enabled* if U cannot be extended by any transition t satisfying $\ell(t) \in \ell(U)$ to yield a step which is free-enabled at M. That is, locally maximal concurrency (lmax-enabledness) is similar to maximal concurrency, but now only active localities cannot execute further transitions. For the PTL-net in Figure 4 we have that $\{\tau_1^{r_{11}}, \tau_1^{r_{13}}\}$ is lmax-enabled at the initial marking, but $\{\tau_1^{r_{11}}\}$ is not. All the remaining notions are the same as for PT-nets.

3 Membrane Systems and Petri Nets

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), as shown in Figure 2.

Let V be a finite alphabet of names of *objects* (molecules). A *basic membrane system* over μ is a tuple

$$\Pi = (V, \mu, w_1^0, \ldots, w_m^0, R_1, \ldots, R_m)$$

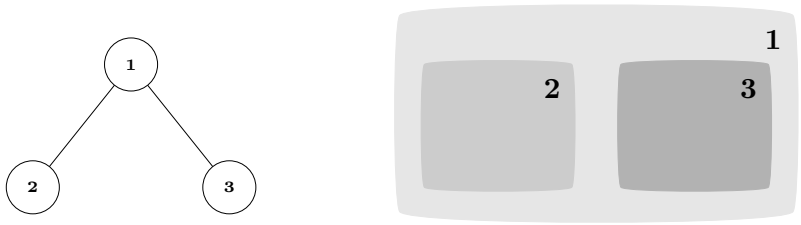

Figure 2: A membrane structure ($m = 3$) and its compartments with 1 being the root node, $(1, 2) \in \mu$ and $1 = \mathit{parent}(3)$.

such that, for every membrane i, w_i^0 is a multiset of objects, and R_i is a finite set of *evolution rules* r of the form $\mathit{lhs}^r \to \mathit{rhs}^r$, where lhs^r (the left hand side of r) is a non-empty multiset over V, and rhs^r (the right hand side of r) is a non-empty multiset over

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

Note that a symbol a_{in_j} represents an object a that is sent to a child node (compartment) j and a_{out} means that a is sent to the parent node. If i is the root of μ then no indexed object of the form a_{out} belongs to rhs^r. A *configuration* of Π is a tuple

$$C = (w_1, \ldots, w_m)$$

of multisets of objects, and $C_0 = (w_1^0, \ldots, w_m^0)$ is the *initial* configuration. Figure 3 shows a basic membrane system over the membrane structure from Figure 2.

A membrane system evolves from configuration to configuration as a consequence of the application (or execution) of evolution rules. There is more than one strategy in which this can be done. Maximal concurrency used to be the standard execution mode for membrane systems. Later, however, also in view of the intrinsic connections with Petri nets, other execution modes attracted interest. In particular, with the concept of localities added to PT-nets to represent compartments, locally maximal concurrency came to light as a new realistic execution semantics for membrane systems. Hence, similarly as in the case of PTL-nets, we can distinguish four such *execution modes*, all based on the notion of a vector multi-rule.

A *vector multi-rule* of Π is a tuple

$$\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$$

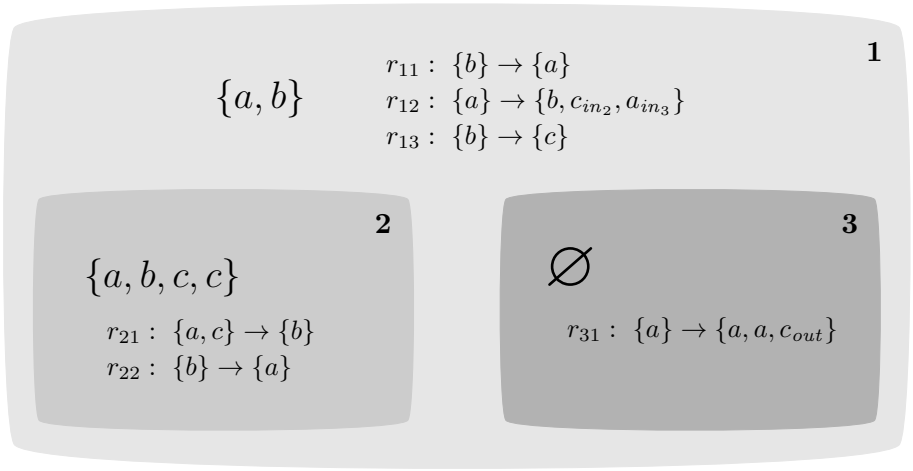

Figure 3: A basic membrane system Π_0.

where, for each membrane i of μ, $\mathbf{r}_i$ is a multiset of rules from R_i. For such a vector multi-rule, we denote by $lhs_i^{\mathbf{r}}$ the multiset

$$\sum_{r \in R_i} \mathbf{r}_i(r) \cdot lhs^r$$

in which all objects in the left hand sides of the rules in $\mathbf{r}_i$ are accumulated, and by $rhs_i^{\mathbf{r}}$ the multiset

$$\sum_{r \in R_i} \mathbf{r}_i(r) \cdot rhs^r$$

of all (indexed) objects in the right hand sides. The first multiset specifies how many objects are needed in each compartment for the simultaneous execution of all the instances evolution rules in $\mathbf{r}$.

A vector multi-rule $\mathbf{r}$ of Π is *free-enabled* at a configuration C if $lhs_i^{\mathbf{r}} \leq w_i$, for each i. A free-enabled $\mathbf{r}$ is: *min-enabled* if $|\mathbf{r}_1| + \cdots + |\mathbf{r}_m| = 1$; *max-enabled* if no $\mathbf{r}_i$ can be extended to yield a vector multi-rule which is free-enabled at C; and *lmax-enabled* if no non-empty $\mathbf{r}_i$ can be extended to yield a vector multi-rule which is free-enabled at C. For example, in Figure 3,

- $\langle \varnothing, \varnothing, \{r_{31}\} \rangle$ is not free-enabled;
- $\langle \{r_{11}\}, \varnothing, \varnothing \rangle$ is min-enabled but not lmax-enabled;
- $\langle \{r_{11}, r_{12}\}, \varnothing, \varnothing \rangle$ is lmax-enabled but not max-enabled; and

- $\langle\{r_{11}, r_{12}\}, \{r_{21}, r_{22}\}, \varnothing\rangle$ is max-enabled.

If $\mathbf{r}$ is free-enabled (*free*) at a configuration C, then C has in each membrane i enough copies of objects for the application of the multiset of evolution rules $\mathbf{r}_i$. Maximal concurrency (*max*) requires that adding any extra rule makes $\mathbf{r}$ demand more objects than C can provide. Locally maximal concurrency (*lmax*) is similar but in this case only those compartments which have rules in $\mathbf{r}$ cannot enable even more rules; in other words, each compartment either uses no rule, or uses a maximal multiset of rules. Minimal enabling (*min*) allows only a single copy of just one rule to be applied any time.

The effect of the application of the rules is independent of the mode of execution $\mathfrak{m} \in \{free, min, max, lmax\}$. A vector multi-rule $\mathbf{r}$ which is $\mathfrak{m}$-enabled at C can $\mathfrak{m}$-*evolve* to a configuration $C' = (w'_1, \ldots w'_m)$ such that, for each i and object a:

$$w'_i(a) = w_i(a) - lhs^{\mathbf{r}}_i(a) + rhs^{\mathbf{r}}_i(a) + rhs^{\mathbf{r}}_{parent(i)}(a_{in_i}) + \sum_{i=parent(j)} rhs^{\mathbf{r}}_j(a_{out})$$

where $rhs^{\mathbf{r}}_{parent(i)} = \varnothing$ if i is the root of μ. We denote this by $C \xrightarrow{\mathbf{r}}_{\mathfrak{m}} C'$. Moreover, an $\mathfrak{m}$-*computation* is a sequence of $\mathfrak{m}$-evolutions starting from the initial configuration. For the example in Figure 3 we have:

$$C_0 \xrightarrow{\mathbf{r}}_{\mathfrak{m}} (\{a, b\}, \{a, b, c, c, c\}, \{a\}) \ ,$$

where $\mathbf{r} = \langle\{r_{11}, r_{12}\}, \varnothing, \varnothing\rangle$.

3.1 Petri Net Modelling of Membrane Systems

There is a natural way of translating a basic membrane system Π into a behaviourally equivalent PTL-net $PTL(\Pi) = (P, T, W, \ell, M_0)$, where multisets of places are used to represent the availability of molecules within the compartments, and transitions correspond to evolution rules. Each transition is associated with a compartment and this information is represented by the localities of net transitions. The constructed PTL-net $PTL(\Pi)$ has a separate place π^a_j for each molecule a and membrane j, and a separate transition τ^r_i with locality i for each rule r in compartment i. The initial marking inserts $w_j(a)$ tokens into each place π^a_j. The connectivity between transition $t = \tau^r_i$ and place $p = \pi^a_j$ is given by:

$$W(p,t) = \begin{cases} lhs^r(a) & \text{if } i = j \\ 0 & \text{otherwise .} \end{cases}$$

as well as:

$$W(t,p) = \begin{cases} rhs^r(a) & \text{if } i = j \\ rhs^r(a_{out}) & \text{if } j = parent(i) \\ rhs^r(a_{in_j}) & \text{if } i = parent(j) \\ 0 & \text{otherwise .} \end{cases}$$

Figure 4 shows the result of the above translation for the basic membrane system in Figure 3.

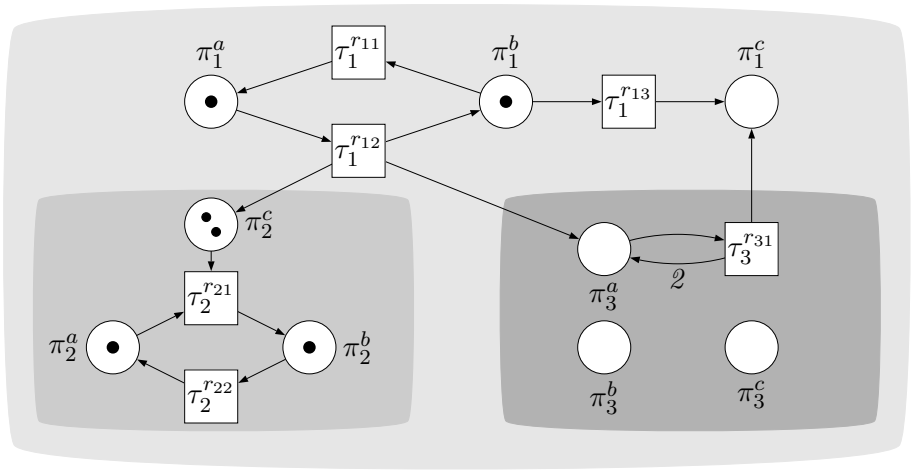

Figure 4: PTL-net $PTL(\Pi_0)$ modelling the basic membrane system Π_0.

The PTL-net $PTL(\Pi)$ provides a *faithful* representation of the behaviour of the basic membrane system Π. To capture this very close relationship, we define two bijective mappings, ν and ρ, which allow us to move between Π and $PTL(\Pi)$:

- For every marking M of $PTL(\Pi)$, $\nu(M) = (w_1, \ldots, w_m)$ is the configuration of Π, given by $w_i(a) = M(\pi_i^a)$, for every molecule a.

- For every step U of $PTL(\Pi)$, $\rho(U) = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$ is the vector multi-rule of Π, given by $\mathbf{r}_i(r) = U(\tau_i^r)$, for every compartment i and every rule $r \in R_i$.

It is then possible to establish a direct relationship between (the operation of) the original membrane system and the PTL-net resulting from the above

translation at the system level:

$$\begin{array}{lcl} C \xrightarrow{\mathbf{r}}_{\mathfrak{m}} C' & \Longrightarrow & \nu^{-1}(C)\ [\rho^{-1}(\mathbf{r})\rangle_{\mathfrak{m}}\ \nu^{-1}(C') \\ M[U\rangle_{\mathfrak{m}} M' & \Longrightarrow & \nu(M)\ \xrightarrow{\rho(U)}_{\mathfrak{m}}\ \nu(M') \end{array} \tag{1}$$

for all modes of execution $\mathfrak{m} \in \{free, min, max, lmax\}$, configurations C of Π and markings M of $PTL(\Pi)$. Together with $\nu(M_0) = C_0$, such a result means that the (finite and infinite) $\mathfrak{m}$-step sequences of $PTL(\Pi)$ faithfully represent $\mathfrak{m}$-computations of Π, and the same applies to markings and configurations.

3.2 Petri Net Analysis of Membrane Systems

Thanks to the very tight behavioural correspondence between Π and $PTL(\Pi)$ captured by (1) above, analytical techniques developed for Petri nets can be applied to membrane systems. For example, one can use the invariant analysis based on linear algebra [42] to verify properties of configurations reachable from the initial one. If we take the membrane system in Figure 3 and apply the invariant analysis to the corresponding Petri net in Figure 4, then one can deduce that the total number of molecules a and b in compartment **2** is constant, irrespective of the initial configuration.

Another direction is to use the causality semantics approach of Petri nets based on occurrence nets, allowing one to analyse entire computations rather than individual reachable configurations. In particular, occurrence nets allow one to investigate causality, concurrency and executability in system behaviour. In case of lmax-step sequences, however, one needs to modify the standard process construction, as first outlined in [26].

To analyse the state space of Π one can also employ the reachability graph of $PTL(\Pi)$. Investigating reachability graphs has a long tradition in the field of Petri nets, and has produced several fundamental results. For example, *reachability* for PT-nets is decidable [31, 29] which means that the problem of deciding whether a basic membrane system has a free- or min-execution leading to a given configuration can be decided, even when there are infinitely many reachable configurations. Another relevant property of Π is whether the concentration of specific molecule(s) in specific compartment(s) can grow unboundedly. This problem, known in the are of Petri nets as *boundedness*, can be tackled using the coverability tree construction [18]. Coverability trees can also be used to decide whether two specific molecules can ever be simultaneously present in the same compartment.

4 Reaction Systems and Petri Nets

Reaction systems [10, 11, 12] are a formal framework for the investigation of processes carried out by biochemical reactions. Thus the framework is inspired by biochemistry and its underlying ideas are motivated by the facilitation/acceleration and inhibition/retardation, properties shared by a great number of biochemical reactions. Reaction systems constitute a computational approach inspired by nature and are targeted at the investigation of ongoing dynamic changes occurring in biochemical systems through information processing. However, the model is based on principles remarkably different from those underlying other existing models of computation.

A *reaction system* is a pair

$$\mathcal{A} = (S, A) \ ,$$

where S is a finite *background* set comprising the *entities* of $\mathcal{A}$, and A is the set of reactions of $\mathcal{A}$. Each *reaction* is a triple of the form $a = (R, I, P)$, where the three components are finite sets:[4]

- $R \subseteq S$ is the set of *reactants*;
- $I \subseteq S$ is the set of *inhibitors*; and
- $P \subseteq S$ is the set of *products*.

The components of a reaction $a = (R, I, P)$ may be denoted, respectively, by R_a, I_a and P_a.

A *state* of a reaction system is any set C of its entities. Then an *initialised* reaction system is a triple

$$\mathcal{A} = (S, A, C_0) \ ,$$

where (S, A) is a reaction system and $C_0 \subseteq S$ is the *initial* state. A reaction system with background set S has exactly $2^{|S|}$ potential states. To describe possible moves between these states, we need to say what is meant by an occurrence of a reaction or a set of reactions.

A reaction a is *enabled* at a state $C \subseteq S$ if $R_a \subseteq C$ and $I_a \cap C = \varnothing$; moreover, for the purpose of establishing the relationship with Petri nets, in this paper we will say that a set $\mathcal{R}$ of reactions is enabled at C if each

[4] In the original definition these sets are assumed to be non-empty and $R \cap I = \varnothing$.

reaction of $\mathcal{R}$ is enabled. In such a case, $\mathcal{R}$ can *occur* with its effect on C being given by

$$res_{\mathcal{R}}(C) = \bigcup_{a \in \mathcal{R}} P_a \ .$$

We denote the resulting *state change* by $C \xrightarrow{\mathcal{R}} res_{\mathcal{R}}(C)$. If $\mathcal{R}$ is the set of all reactions enabled at C, then we may simply write $C \longrightarrow C'$, where $C' = res_{\mathcal{R}}(C)$.

In the state change as described above, all the entities in $C \setminus \bigcup_{a \in \mathcal{R}} P_a$ disappear when $\mathcal{R}$ occurs. As a result, and unlike in other formal models of dynamic systems, there is no persistency in a reaction system in the sense that an entity present in a state disappears unless it is sustained by at least one reaction in $\mathcal{R}$. Consider, for instance, an initialised reaction system

$$\mathcal{A}_0 = (\{q, r, s\}, \{a, b, c\}, \{q\}) \ ,$$

with background set $\{q, r, s\}$, the initial state $\{q\}$, and three reactions:

$$a = (\{r, q\}, \varnothing, \{r\}) \qquad b = (\{q\}, \{s\}, \{r, q\}) \qquad c = (\{q\}, \varnothing, \{s\}) \ .$$

Then we have the following examples of state change:

$$\{r, q, s\} \xrightarrow{\{a,c\}} \{r, s\} \qquad \{r, q\} \xrightarrow{\{b\}} \{r, q\} \qquad \{q, r, s\} \xrightarrow{\{a\}} \{r\} \ .$$

One may observe that there is no conflict between reactions in the sense that the occurrence of one reaction might imply that another reaction which is also enabled at the current state, cannot occur. This, again, is a feature not found in most other formal models of dynamic systems.

It is crucial to point explicitly to the 'non-counting' features of reaction systems: entities are either present or not, and produced or not, and reactions can or cannot occur given the presence or absence of certain entities. There is no representation of multiple instances of entities or multiple occurrences of reactions.

In general, reaction systems may have an environment and then operate within a changing context (with entities coming from the environment at each stage of evolution). Here, however, we will consider *context-independent* processes defined by a reaction system with an initial state provided by the environment, and every next state obtained as the result of reactions taking place in the previous state.

4.1 Set-Nets

In [27] we investigate how to construct Petri net representations of reaction systems. While doing so, we made some general observations and assumptions about the relationship between reaction systems and nets.

- Entities can be represented by places, and reactions by net transitions.
- Since there are no conflicts between reactions, activator arcs can be used to test for the presence of reactants (rather than claiming resources for the exclusive use as with ordinary arcs and input places).
- Inhibitor arcs can be used to test for the absence of reactants.
- All reactions that can occur in a reaction system do occur, and the only entities left after a state change are the newly generated products. In the Petri net framework, these features correspond to *maximal concurrency* and *place resetting* implemented by reset arcs [9].

We tried four different modelling methods, including high-level Petri nets [17]. In each case, we established a close correspondence between the evolutions of two corresponding models. All these net models, however, exhibited deficiencies w.r.t. simplicity and/or elegance and/or tractability of the translation.

In particular, in all four cases, one state of a reaction system would correspond to many markings of a corresponding Petri net, which is dramatically different from the one-to-one relationship between the configurations of a membrane systems and markings of a the corresponding PTL-net described in Section 3. We therefore proposed a new class of Petri nets, called SET-nets, which provide a stronger match with reaction systems and their semantics.

The main idea is that in a SET-net there is no concept of counting. Places are marked or not marked and arcs have no weights. Set-nets resemble elementary net systems (EN-systems) [38] which is a fundamental model to study basic features of concurrent systems, including conflict, causality and independence. However, their execution semantics is different. In SET-nets, a marked place indicates the presence of a resource without any quantification. Hence any number of transitions that take input from this place can be fired at the same time. Moreover, firing a transition empties all its input places. Thus there are no conflicts over tokens in SET-nets, unlike in EN-systems or PT-nets. Similarly, places do not count the tokens, and the firing of a transition simply marks each of its output places (whether or not

they were already marked). We will build up the new model in two stages, introducing first SET-nets with only flow arcs.

A SET-*net* is a tuple

$$SN = (Pl, Tr, W, M_0)$$

such that the four components are as in the definition of PT-nets, under the proviso that W always returns 0 or 1, and the initial marking M_0 is a set (in general, *markings* are now sets of places). The firing rule for SN assumes that each step U is a set, and we denote by ${}^\bullet U$ and $U^\bullet$ the sets of all places p such that there is a transition $t \in U$ with $W(p,t) = 1$ and $W(t,p) = 1$, respectively. We then say that U is *enabled* at a marking M if ${}^\bullet U \subseteq M$. In such a case, U can be *executed* with its effect on M being given by the resulting marking

$$M' = (M \setminus {}^\bullet U) \cup U^\bullet \ .$$

We denote this by $M[U\rangle M'$. Moreover, if U is the set of all transitions enabled at M (i.e., all transitions t such that $\{t\}$ is enabled), then we obtain a maximally concurrent execution denoted by $M[U\rangle_{max} M'$ or simply $M[\rangle_{max} M'$ as the maximally concurrent step enabled at M is unique. Hence a step U enabled at a marking M may contain two distinct transitions t and u for which ${}^\bullet t \cap {}^\bullet u \neq \varnothing$ or $t^\bullet \cap u^\bullet \neq \varnothing$ and yet the common places will never contain more than one token.

To model reaction systems, we need additionally inhibitor arcs to capture the effect of inhibitors in reaction rules. We therefore consider SET-nets with *inhibitor arcs* $Inh \subseteq Pl \times Tr$. In such a case, the enabling relation changes similarly as for PT-nets with inhibitor arcs, and we say that a step U is *enabled* at a marking M if

$${}^\bullet U \subseteq M \quad \text{and} \quad (M \times U) \cap Inh = \varnothing \ .$$

The result of executing an enabled step remains the same as before.

4.2 SET-Nets Modelling of Reaction Systems

Reaction systems and SET-nets fit together well in the sense that both do not count tokens and both change states on the basis of the presence/absence of resources, represented by sets. Moreover, under the SET-net semantics, ordinary arcs (transitions) can be used to empty places. Finally, following the assumption that all reactions that can take place do take place, the maximal set-semantics can be employed.

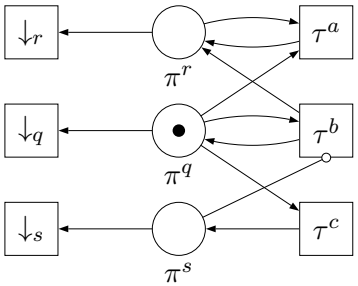

Figure 5: SET-net $SN(\mathcal{A}_0)$ modelling reaction system $\mathcal{A}_0$.

Modelling inhibition aspects of reactions is rather straightforward, as illustrated by the SET-net in Figure 5, representing the context-independent initialised reaction system $\mathcal{A}_0$ considered earlier. As we already mentioned, places represent entities. Transitions τ^a, τ^b and τ^c represent reactions, and $r_\downarrow$, $q_\downarrow$ and $s_\downarrow$ ensure that once the SET-net is active only tokens produced in the last maximal step are present in the current marking. Using inhibitor arcs gives a compact translation of reaction systems which is in a sense minimal w.r.t. the number of places, arcs and transitions. Moreover, relating the behaviour of the resulting SET-nets and the original reaction systems can be done as before.

Formally, the places, transitions and initial marking of the translation $SN(\mathcal{A})$ are given by:

$$\begin{array}{lcl} Pl & = & \{\pi^s \mid s \in S\} \\ Tr & = & \{\tau^a \mid a \in A\} \cup \{\downarrow_s \mid s \in S\} \\ M_0 & = & \{\pi^s \mid s \in C_0\}\ . \end{array}$$

The flow and inhibitor arcs are as follows:

$$\begin{array}{lcl} W & = & \{(\pi^s, \downarrow_s) \mid s \in S\}\ \cup \\ & & \{(\pi^s, \tau^a) \mid a \in A \wedge s \in R_a\}\ \cup \\ & & \{(\tau^a, \pi^s) \mid a \in A \wedge s \in P_a\} \\ Inh & = & \{(\pi^s, \tau^a) \mid a \in A \wedge s \in I_a\}\ . \end{array}$$

Relating the behaviour of the SET-net model $SN(\mathcal{A})$ and the original reaction system $\mathcal{A}$ is straightforward using the mappings $\nu(M) = \{s \mid \pi^s \in M\}$, for every marking $M \subseteq Pl$, and $\rho(U) = \{a \mid \tau^a \in U\}$, for every step $U \subseteq Tr$.

It is then possible to establish a direct relationship between (the oper-

ation of) reaction systems and SET-nets at the system level:

$$\begin{array}{lcl} C \xrightarrow{\mathcal{R}} C' & \Longrightarrow & \exists U:\ \rho(U) = \mathcal{R} \wedge \nu^{-1}(C)\ [U\rangle\ \nu^{-1}(C') \\ M[U\rangle M' & \Longrightarrow & \nu(M) \xrightarrow{\rho(U)} \nu(M') \end{array} \tag{2}$$

for each state C of $\mathcal{A}$, and each marking M of $SN(\mathcal{A})$. Together with $\nu(C_0) = M_0$, such a result means that the (finite and infinite) step sequences of $SN(\mathcal{A})$ faithfully represent computations of $\mathcal{A}$, and that there is a one-to-one correspondence between states and markings.

Moreover, the maximally concurrent semantics of the SET-net corresponds to the execution of the reaction system, i.e., the execution of all enabled reactions in each evolution step:

$$\begin{array}{lcl} C \longrightarrow C' & \Longrightarrow & \nu^{-1}(C)\ [\rangle_{max}\ \nu^{-1}(C') \\ M[\rangle_{max} M' & \Longrightarrow & \nu(M) \longrightarrow \nu(M') \end{array}$$

for each state C of $\mathcal{A}$, and each marking M of $SN(\mathcal{A})$.

Note that the fundamental class of EN-systems [38] extended with inhibitor as well as activator arcs [16, 30, 32] basically has the same static structure as SET-nets. However, their treatment of conflicts between transitions accessing the same token, as well blocking a transition which could add a token to a marked place, are totally different. The latter issue has been noted in the past, and the constraint relaxed. For example, there are variations of Petri nets, such as Boolean Petri nets, where adding a token to an already marked place does not add another token [5, 6, 14]. Also, behaviour of this kind was mentioned in [2] in the context of net synthesis. Having said that, the semantics considered in prior works was based on single transition firings, rather than (maximal) steps as is the case for SET-nets, and so the issue of 'token sharing' was never explicitly considered.

The main initial motivation of our investigation was to see how Petri net based concepts could be deployed to analyse reaction systems. In doing so, we introduced the model of SET-nets which is an original contribution to the field of Petri nets. In the next section we will see how the introduction of SET-nets has motivated the introduction of a new model of membrane systems.

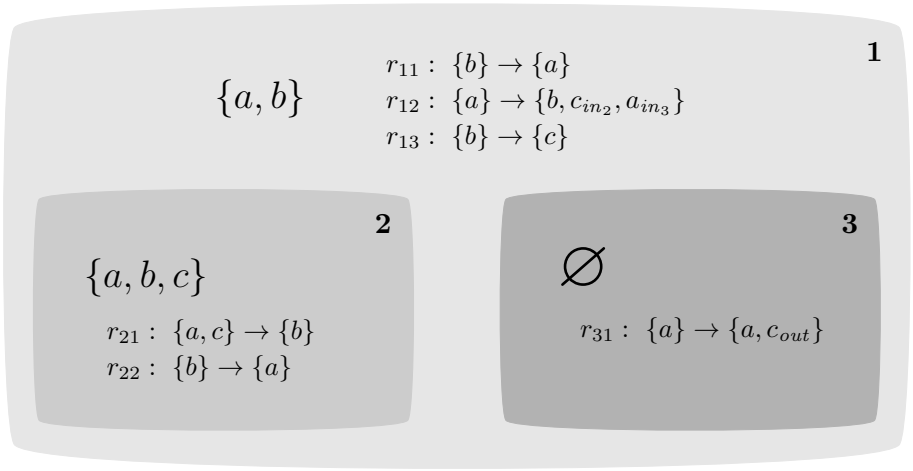

Figure 6: A basic set membrane system Σ_0.

5 Petri Nets and Set Membrane Systems

We now discuss membrane systems which use 'qualitative' rather than 'quantitative' application of evolution rules to change the current state. The formal definitions and representation for this class of membrane are as those in Section 3, except that we are now working with sets rather than multisets of objects and evolution rules (similar to the operation of membrane systems in [1] where a qualitative approach was used in the application of rules within membranes, but a quantitative on for sending objects to the environment).

A *basic set membrane system* over the membrane structure μ is a tuple

$$\Sigma = (V, \mu, w_1^0, \ldots, w_m^0, R_1, \ldots, R_m)$$

as in Section 3, where each w_i^0 is a set of objects, and the left and right hand sides of every evolution rule are non-empty sets. Similarly, each configuration is composed of sets of objects.

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 $\mathbf{r} \leq \mathbf{r}'$ if $\mathbf{r}_i \subseteq \mathbf{r}'_i$, for each $i \leq m$; and $\mathbf{r} < \mathbf{r}'$ if $\mathbf{r} \subseteq \mathbf{r}'$ and $\mathbf{r} \neq \mathbf{r}'$. 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. We then say that a vector set-rule $\mathbf{r}$ is free-enabled / min-enabled / max-enabled / lmax-enabled exactly as in Section 3. Following this, 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:

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

We denote this by $C \xrightarrow{\mathbf{r}}_{\mathfrak{m}} C'$. 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. 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.

5.1 SET-Nets Modelling of Set Membrane Systems

To faithfully capture the behaviour of basic set membrane systems, we need to extend SET-nets with localities. A SET-*net with localities* (or SETL-*net*) *SNL* is a SET-net together with a locality mapping $\ell : Tr \to \mathbb{N}$ as in PTL-nets. Based on the semantics of the underlying SET-net and the semantics of nets with localities, one can then introduce four modes of execution, free-enabled / min-enabled / max-enabled / lmax-enabled, in a straightforward way.

The modelling of a basic set membrane system Σ as a SETL-net $SNL(\Sigma)$ follows exactly the same lines as in Section 3 (note that all arcs have weight 1 in this case, and there are no inhibitor nor activator arcs). Figure 7 shows the translation for the basic set membrane system Σ_0 in Figure 6.

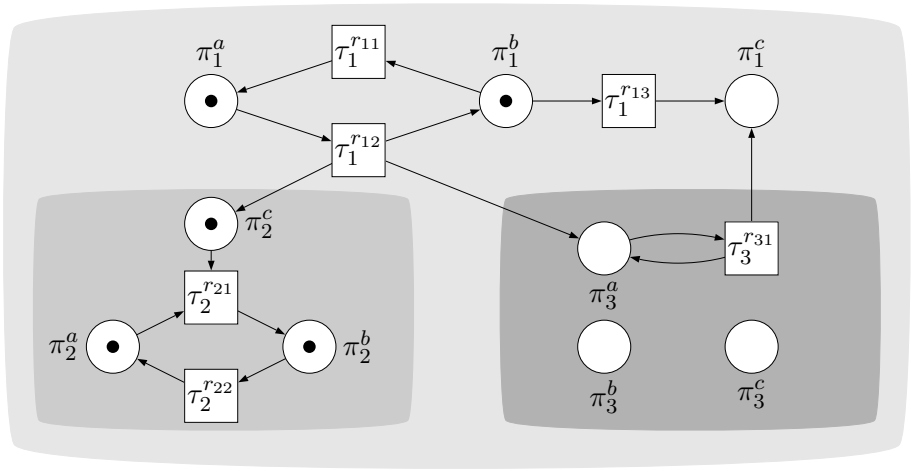

Figure 7: SETL-net $SNL(\Sigma_0)$ modelling the basic set membrane system Σ_0.

The tight correspondence between the membrane system Σ and the SETL-net $SNL(\Sigma)$ is captured by the same two bijective mappings ν and ρ, now restricted to sets, as in Section 3. Moreover, the key result (1) establishing the faithfulness of the translation obtained there can be re-stated for the current translation. This, in particular, means that the (finite and infinite) $\mathfrak{m}$-step sequences of $SNL(\Sigma)$ faithfully represent $\mathfrak{m}$-computations of Σ.

5.2 Petri Net Analysis of Set Membrane Systems

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.

In [25], we have already made preliminary investigation into the *synthesis problem* which aims at an automatic construction of SET-nets exhibiting 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.

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 SETL-nets) as a language. These languages are regular subset languages. The study of subset languages of Petri nets was initiated in [40, 41] but still for the standard (quantitative) interpretation. There are a number of interesting theoretical questions and topics for the regular subset languages generated by SETL-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 different objects (types of molecules) to appear in a given compartment at some point together.

6 Petri Nets and Extended Membrane Systems

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 translations to Petri nets (see, for example, [15]). 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., [20]).

First we consider an extended version of membrane systems for modelling how reactions may be triggered or blocked in the presence of certain molecules. The role of such molecules differs from that of catalysts which actively take part in reactions and are returned afterwards. An example is object a which acts as a catalyst in evolution rule

$$r : \{a, b\} \to \{a, b_{out}\} \ .$$

To model the subtle effect that the presence of molecules may have, membrane systems have evolution rules r of the form

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

where pro^r and inh^r are multisets over V specifying respectively the *promoters* and *inhibitors*. The intuition behind pro^r and inh^r is that they test respectively for a minimal or maximal number of certain objects inside a compartment, but without consuming them. As a consequence, any number of rules can test for the presence of a single object at the same time. In order for r to occur there must be *at least* $pro^r(a)$ copies of each symbol a in its associated compartment, and *less than* $inh^r(a)$ copies of each symbol a which occurs in inh^r. Thus we retain all definitions introduced for basic membrane systems with only one change regarding the notion of a free-enabled vector multi-rule $\mathbf{r}$. This is strengthened by additionally requiring that, for each i and $r \in \mathbf{r}_i$, we have $pro_i^{\mathbf{r}} \leq w_i$ and, moreover, if $a \in inh^r$ then $w_i(a) < inh^r(a)$.

PTL-nets are not expressive enough to model inhibitors and promoters because arcs between transitions and places indicate consumption and production of tokens (objects) rather than testing for their presence or absence. A possible way out is to use PTL-nets extended with *range* arcs [19]. Each such arc links a place to a transition and is specified by a closed interval (possibly infinite) of non-negative integers. This interval indicates the *range* (a closed interval of natural numbers) for the number of tokens that should be present in the place to enable the occurrence of the transition. Clearly, like pro^r and inh^r, range arcs can be used to model certain forbidden/required concentrations of molecules in a compartment.

In [20], it has been shown that key properties of the modified translation are very similar to those obtained in the basic case; in particular, the correspondence result (1) can simply be restated. Moreover, the treatment of causality developed for PTL-nets can also be extended (see [20]).

What may come as a surprise, is that PTL-nets with inhibitor and activator arcs are also robust enough to model in a faithful way, membrane systems which have a dynamic structure due to rules which may thicken or dissolve membranes [21].

When it comes to the properties which might be expressed or investigated using reachability or coverability graphs, the situation changes dramatically when we move from PTL-nets to PTL-nets with range arcs. The reason is that the latter allow one to test for the absence of resources (zero-testing).

Nets with this kind of relationship between places and transitions (i.e., inhibitor arcs) have been considered in [13]. The extension with inhibitor arcs gives the resulting model of Petri nets the expressive power of Turing machines; net languages become recursively enumerable rather than recursive, and decidability for certain important behavioural properties, such as reachability and boundedness, is lost [13], partial solutions have been proposed by restricting the class of nets under consideration as, for example, in [4, 22].

7 Conclusion

In this paper we described a number of results obtained while working at the interface between Petri nets (by now a classical formal model for dealing with distributed systems), and two recently proposed formal approaches aimed at dealing with computations inspired by biochemical reactions (i.e., membrane systems and reaction systems). Our overall experience was both illuminating and highly encouraging. We have found that the different models share a number of important features which allowed us to compare them in a meaningful way (see, e.g., the faithfulness results (1) and (2)), but at the same time the original models did differ in some crucial aspects. The latter realisation has provided a direct motivation for importing concepts, such as the structuring of the molecules in a membrane system into compartments and the qualitative treatment of entities in reaction systems, into the realm of Petri nets. This has resulted in the introduction of new net classes, such as PTL-nets and SET-nets, which should be of a general interest to various application oriented research communities. For instance, nets with localities together with the lmax-semantics are relevant for dealing with globally asynchronous locally synchronous (or GALS) systems. Moreover, we transferred newly defined modelling concepts back into the domain of membrane systems, by developing set membrane systems. We therefore experience a harmonious synergy of the three, originally separate, models of computation.

Results outlined in this paper open up a way to the adoption of Petri net analytical techniques in the areas of membrane computing and reaction systems. For example, one of the key advantages of Petri nets is that they support in a clear and unambiguous way concepts relating to causality and concurrency, and it turns out that the Petri net treatment of these concepts can be extended to membrane systems thanks to the so-called barb events which capture the intricacies of the lmax-semantics. We therefore feel that Petri nets are a robust model which can be suitably extended to provide

valuable insights into other, seemingly distant approaches.

References

[1] A.Alhazov. P Systems Without Multiplicities of Symbol-objects. *Information Processing Letters* 100:124–129, 2006

[2] E.Badouel and P.Darondeau. Theory of Regions. volume 1491 of *Lecture Notes in Computer Science*, pages 529–586, Springer-Verlag, 1998

[3] R.Brijder, A.Ehrenfeucht, M.G.Main and G.Rozenberg. A Tour of Reaction Systems. *Int. Journal of Foundations of Computer Science*, 2011

[4] N.Busi. Analysis Issues in Petri Nets with Inhibitor Arcs. *Theoretical Computer Science* 275:127–177, 2002

[5] L.Czaja. A Calculus of Nets. *Cybernetics and Systems Analysis* 29:185–193, 1993

[6] P.De Bra, G.J.Houben and Y.Kornatzky. A Formal Approach to Analyzing the Browsing Semantics of Hypertext. *Proc. CSN-94 Conference*, pages 78–89, 1994

[7] J.Desel and G.Juhas. What Is a Petri Net? volume 2128 of *Lecture Notes in Computer Science*, pages 1–25, Springer-Verlag, 2001

[8] J.Desel and W.Reisig. volume 1491 of *Lecture Notes in Computer Science*, pages 122–173, Springer-Verlag, 1998

[9] C.Dufourd, A.Finkel and Ph.Schnoebelen. Reset Nets Between Decidability and Undecidability. volume 1443 of *Lecture Notes in Computer Science*, pages 103–115, Springer-Verlag, 1998

[10] A.Ehrenfeucht, M.Main and G.Rozenberg. Combinatorics of Life and Death for Reaction Systems. *International Journal of Foundations of Computer Science* 22:345–356, 2009

[11] A.Ehrenfeucht and G.Rozenberg. Reaction Systems. *Fundamenta Informaticae* 76:1–18, 2006

[12] A.Ehrenfeucht, M.Main and G.Rozenberg. Events and Modules in Reaction Systems. *Theoretical Computer Science* 376:3–16, 2007

[13] M.Hack. Decision Problems for Petri Nets and Vector Addition Systems. *Technical Memo* 59, Project MAC, MIT, 1975

[14] M.Heiner, D.Gilbert and R.Donaldson. Petri Nets for Systems and Synthetic Biology. volume 5016 of *Lecture Notes in Computer Science*, pages 215–264, Springer-Verlag, 2008

[15] O.H.Ibarra, Z.Dang and O.Egecioglu. Catalytic P Systems, Semilinear Sets, and Vector Addition Systems. *Theoretical Computer Science* 312:379–399, 2004

[16] R.Janicki and M.Koutny. Semantics of Inhibitor Nets. *Information and Computation* 123:1–16, 1995

[17] K.Jensen. Coloured Petri Nets and the Invariant-Method. *Theoretical Compututer Science* 14:317–336, 1981

[18] R.M.Karp and R.E.Miller. Parallel Program Schemata. *J. Comput. Syst. Sci.* 3:147–195, 1969

[19] J.Kleijn and M.Koutny. Processes of Petri Nets with Range Testing. *Fundamenta Informaticae* 80:199–219, 2007

[20] J.Kleijn and M.Koutny. Processes of Membrane systems with Promoters and Inhibitors. *Theoretical Computer Science* 404:112–126, 2008

[21] J.Kleijn and M.Koutny. A Petri Net Model for Membrane Systems with Dynamic Structure. *Natural Computing* 8:781–796, 2009

[22] J.Kleijn and M.Koutny. Step Coverability Algorithms for Communicating Systems. *Science of Computer Programming*, 2010, doi:10.1016/j.scico2010.11.003

[23] J.Kleijn and M.Koutny. Petri Nets and Membrane Computing. In: [36], pages 389–412, 2010

[24] J.Kleijn and M.Koutny. Membrane systems with Qualitative Evolution Rules. *Fundamenta Informaticae*, to appear (2011)

[25] J.Kleijn, M.Koutny, M.Pietkiewicz-Koutny and G.Rozenberg. Classifying Boolean Nets for Region-based Synthesis. *CEUR Workshop Proceedings* 725, pages 5–21, 2011

[26] J.Kleijn, M.Koutny and G.Rozenberg. Process Semantics for Membrane Systems. *Journal of Automata, Languages and Combinatorics* 11:321–340, 2006

[27] J.Kleijn, M.Koutny and G.Rozenberg. Modelling Reaction Systems with Petri Nets. *CEUR Workshop Proceedings* 724, pages 36–52, 2011

[28] I.Koch, W.Reisig and F.Schreiber. *Modeling in Systems Biology — The Petri Net Approach*. Springer London Dordrecht Heidelberg New York, 2011

[29] S.R.Kosaraju. Decidability of Reachability in Vector Addition Systems. *Proc. of STOC'82*, pages 267–281, ACM, 1982

[30] M.Koutny and M.Pietkiewicz-Koutny. Synthesis of Elementary Net Systems with Context Arcs and Localities. *Fundamenta Informaticae* 88:307–328, 2008

[31] E.W.Mayr. An Algorithm for the General Petri Net Reachability Problem. *SIAM J. Comput.* 13:441–460, 1984

[32] U.Montanari and F.Rossi. Contextual Nets. *Acta Informatica* 32:545–596, 1995

[33] G.Păun. Computing with Membranes. *J. Comput. Syst. Sci.* 61:108–143, 2000

[34] G.Păun. *Membrane Computing, An Introduction.* Springer-Verlag, Berlin Heidelberg New York, 2002

[35] G.Păun and G.Rozenberg. A Guide to Membrane Computing. *Theoretical Computer Science* 287:73–100, 2002

[36] G.Păun, G.Rozenberg and A.Salomaa. *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2009

[37] C.A.Petri. *Kommunikation mit Automaten.* PhD Thesis, 1962

[38] G.Rozenberg and J.Engelfriet. Elementary Net Systems. volume 1491 of *Lecture Notes in Computer Science*, pages 12–121, Springer-Verlag, 1998

[39] W.Reisig and G.Rozenberg (eds.) Lectures on Petri Nets I and II. volumes 1491, 1492 of *Lecture Notes in Computer Science*, Springer-Verlag, 1998

[40] G.Rozenberg and R.Verraedt. Subset Languages of Petri Nets Part I: The Relationship to String Languages and Normal Forms. *Theoretical Computer Science* 26:301–326, 1983

[41] G.Rozenberg and R.Verraedt. Subset Languages of Petri Nets Part II: Closure Properties. *Theoretical Computer Science* 27:85–108, 1983

[42] M.Silva, E.Teruel and J.M.Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems. volume 1491 of *Lecture Notes in Computer Science*, pages 309–373, Springer-Verlag, 1998