Membrane Systems and Petri Net Synthesis (Invited Paper)

Jetty Kleijn
LIACS
Leiden University
Leiden, The Netherlands
kleijn@liacs.nl

Maciej Koutny
School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
maciej.koutny@ncl.ac.uk

Marta Pietkiewicz-Koutny
School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
marta.koutny@ncl.ac.uk

Grzegorz Rozenberg
LIACS
Leiden University
Leiden, The Netherlands
and
Department of Computer Science
University of Colorado at Boulder
Boulder, Colorado, USA
rozenber@liacs.nl

Automated synthesis from behavioural specifications is an attractive and powerful way of constructing concurrent systems. Here we focus on the problem of synthesising a membrane system from a behavioural specification given in the form of a transition system which specifies the desired state space of the system to be constructed. We demonstrate how a Petri net solution to this problem, based on the notion of region of a transition system, yields a method of automated synthesis of membrane systems from state spaces.

1 Introduction

Membrane systems ([19, 20, 21, 22]) are a computational model inspired by the functioning of living cells and their architecture and in particular, the way chemical reactions take place in cells 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. As a result, membrane systems are essentially multiset rewriting systems. The dynamic aspects of the membrane system model including potential behaviour (computations), derive from such evolution rules.

Petri nets (see, e.g., [5, 23, 25]) are a well-established general model for distributed computation with an extensive range of tools and methods for construction, analysis, and verification of concurrent systems. Their diverse applications areas include computational and operational foundations for problems and issues arising in biology; see for example, [15], for a recent comprehensive overview of applications of Petri nets in systems biology.

There are intrinsic similarities between Petri nets and membrane systems. In particular, there exists a canonical way of translating membrane systems into Petri nets. This translation is faithful in the sense that it relates computation steps at the lowest level and induces in a natural way (sometimes new) extensions and interpretations of Petri net structure and behaviour (e.g., inhibitor arcs, localities, and maximal concurrency). More details on the relationship between Petri nets and membrane systems can be found in, e.g., [9, 14].

Gabriel Ciobanu (Ed.): Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2012)
EPTCS 100, 2012, pp. 1–13, doi:10.4204/EPTCS.100.1

The strong semantical link between the two models invites to extend where necessary and possible existing Petri net techniques and bring them to the domain of membrane systems. An example is the process semantics of Petri nets that can help to understand the dynamics and causality in the biological evolutions represented by membrane systems [8, 12]. In this paper, we focus on the synthesis problem, that is, the problem of automated construction of a system from a specification of its (observed or desired) behaviour.

Automated synthesis from behavioural specifications is an attractive and powerful way of constructing correct concurrent systems [1, 2, 4, 6, 7, 18, 24]. Here we will re-visit the problem of synthesising a Petri net from a behavioural specification given in the form of a transition system. The latter specifies the desired state space of the Petri net to be constructed. We will recall a solution to this problem based on the notion of region of a transition system. We will then demonstrate how this solution leads to a method of automated synthesis of basic membrane systems from state spaces. We also discuss how the proposed method could be extended to cope with more complicated kinds of membrane systems.

2 Preliminaries

Multisets. A multiset over a finite set X is a function $\theta : X \to \mathbb{N} = \{0,1,2,\ldots\}$. θ may be represented by listing its elements with repetitions, e.g., $\theta = \{y,y,z\}$ is such that $\theta(y) = 2$, $\theta(z) = 1$, and $\theta(x) = 0$ otherwise. θ is said to be empty (and denoted by $\varnothing$) if there are no x such that $x \in \theta$ by which we mean that $x \in X$ and $\theta(x) \geq 1$.

For two multisets θ and θ' over X, the sum $\theta + \theta'$ is the multiset given by $(\theta + \theta')(x) = \theta(x) + \theta'(x)$ for all $x \in X$, and for $k \in \mathbb{N}$ the multiset $k \cdot \theta$ is given by $(k \cdot \theta)(x) = k \cdot \theta(x)$ for all $x \in X$. The difference $\theta - \theta'$ is given by $(\theta - \theta')(x) = \max\{\theta(x) - \theta'(x), 0\}$ for all $x \in X$. We denote $\theta \leq \theta'$ whenever $\theta(x) \leq \theta'(x)$ for all $x \in X$, and $\theta < \theta'$ whenever $\theta \leq \theta'$ and $\theta \neq \theta'$. The restriction $\theta|_Z$ of θ to a subset $Z \subseteq X$ is given by $\theta|_Z(x) = \theta(x)$ for $x \in Z$, and $\theta|_Z(x) = 0$ otherwise. The size $|\theta|$ of θ is given by $\sum_{x \in X} \theta(x)$. If $f : X \to Y$ is a function then $f(\theta)$ is the multiset over Y such that $f(\theta)(y) = \sum_{x \in f^{-1}(y)} \theta(x)$, for every $y \in Y$.

Step transition systems. A *step transition system* over a finite set (of actions) A is a triple $TS = (Q, \mathscr{A}, q_0)$, where: Q is a set of nodes called *states*; $\mathscr{A}$ is the set of *arcs*, each arc being a triple (q, α, q') such that $q, q' \in Q$ are states and α is a multiset over A; and $q_0 \in Q$ is the *initial* state. We may write $q \xrightarrow{\alpha} q'$ whenever (q, α, q') is an arc, and denote by

$$tsSteps_q = \{\alpha \mid \alpha \neq \varnothing \wedge \exists q' :\ q \xrightarrow{\alpha} q'\}$$

the set of nonempty steps enabled at a state q in TS. We additionally assume that:

- if $q \xrightarrow{\alpha} q'$ and $q \xrightarrow{\alpha} q''$ then $q' = q''$ (i.e., TS is deterministic);
- for every state $q \in Q$, there is a path from q_0 leading to q;
- for every action $a \in A$, there is an arc $q \xrightarrow{\alpha} q'$ in TS such that $a \in \alpha$; and
- for every state $q \in Q$, we have $q \xrightarrow{\varnothing} q'$ iff $q = q'$.

Let $TS = (Q, \mathscr{A}, q_0)$ be a step transition system over a set of actions A, and $TS' = (Q', \mathscr{A}', q_0')$ be a step transition system over a set of actions A'. TS and TS' are *isomorphic* if there are two bijections, $\phi : A \to A'$ and $\nu : Q \to Q'$, such that $\nu(q_0) = q_0'$ and, for all states $q, q' \in Q$ and multisets α over A:

$$(q, \alpha, q') \in \mathscr{A} \iff (\nu(q), \phi(\alpha), \nu(q')) \in \mathscr{A}' .$$

We denote this by $TS \sim_{\phi,\nu} TS'$ or $TS \sim TS'$.

Petri nets. A *Place/Transition net* (or PT-net) is specified as a tuple $PT = (P,T,W,M_0)$, where: P and T are finite disjoint sets of respectively *places* and *transitions*; $W : (T \times P) \cup (P \times T) \to \mathbb{N}$ is the *arc weight function*; and $M_0 : P \to \mathbb{N}$ is the *initial marking* (in general, any multiset of places is a marking). We assume that, for each transition t, there is at least one place p such that $W(p,t) > 0$. In diagrams, such as 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).

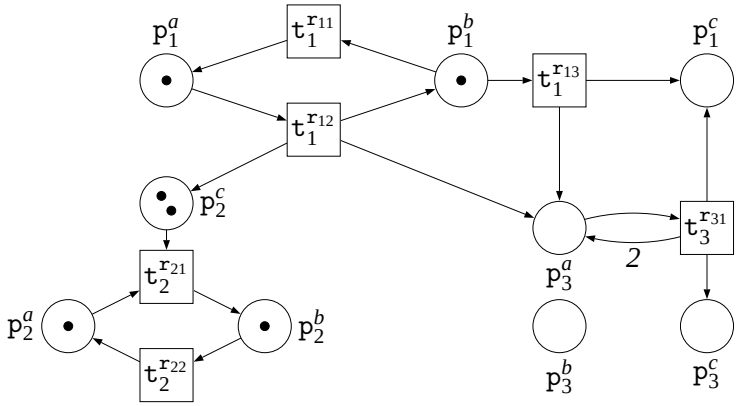

Figure 1: A PT-net.

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:

$$ {}^\bullet\{\mathrm{t}_1^{r_{11}}, \mathrm{t}_1^{r_{11}}, \mathrm{t}_3^{r_{31}}\} = \{\mathrm{p}_1^b, \mathrm{p}_1^b, \mathrm{p}_3^a\} \text{ and } \{\mathrm{t}_1^{r_{11}}, \mathrm{t}_1^{r_{11}}, \mathrm{t}_3^{r_{31}}\}^\bullet = \{\mathrm{p}_1^a, \mathrm{p}_1^a, \mathrm{p}_1^c, \mathrm{p}_3^a, \mathrm{p}_3^a, \mathrm{p}_3^c\}. $$

We distinguish two basic modes of execution of PT-nets. To start with, a step of transitions U is *free-enabled* at a marking M if ${}^\bullet U \leq M$. We denote this by $M[U\rangle_{free}$, and then say that a free-enabled U is *max-enabled* at M if U cannot be extended by a transition to yield a step which is free-enabled at M, i.e., there is no $t \in T$ such that $M[U + \{t\}\rangle_{free}$. We denote this by $M[U\rangle_{max}$. In other words, U is free-enabled at M if in each place there are sufficiently many tokens for the specified multiple occurrence of each of its transitions. Maximal concurrency (max-enabledness) means that extending U would demand more tokens than M supplies. For the PT-net in Figure 1 we have that, at the given marking M_0, the step $\{\mathrm{t}_1^{r_{12}}, \mathrm{t}_2^{r_{21}}\}$ is free-enabled but not max-enabled, and $\{\mathrm{t}_1^{r_{11}}, \mathrm{t}_1^{r_{12}}, \mathrm{t}_2^{r_{21}}, \mathrm{t}_2^{r_{22}}\}$ is max-enabled.

For each mode of execution $\mathfrak{m} \in \{free, max\}$, 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 $M[U\rangle_{\mathfrak{m}} M'$. For the PT-net in Figure 1 we have

$$ M_0[\{\mathrm{t}_1^{r_{12}}, \mathrm{t}_2^{r_{21}}\}\rangle_{free} \{\mathrm{p}_1^b, \mathrm{p}_1^b, \mathrm{p}_2^b, \mathrm{p}_2^b, \mathrm{p}_2^c, \mathrm{p}_2^c, \mathrm{p}_3^a\} \,. $$

Petri nets with localities. PT-nets are a general model of concurrent computation. To capture the compartmentisation of membrane systems, [12] adds explicit localities to transitions. Though not necessary from a modelling point of view, we associate in this paper — only for notational convenience — also each place with a locality.

A PT-*net with localities* (or PTL-net) is a tuple $PTL = (P, T, W, \ell, M_0)$ such that (P, T, W, M_0) is a PT-net, and ℓ is a *location* mapping for the transitions and places. Whenever $\ell(x) = \ell(z)$, we call x and z *co-located*. In diagrams, nodes representing co-located transitions and/or places will be shaded in the same way, as shown in Figure 2.

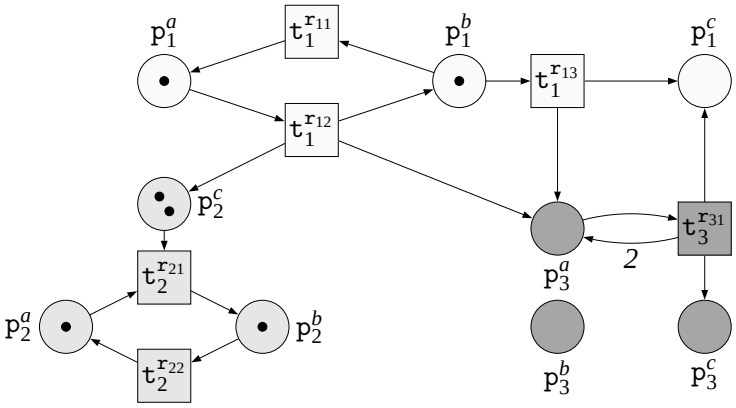

Figure 2: A PTL-net corresponding to a basic membrane system, where $\ell(x_i^z) = i$, for each node of the form x_i^z. Note that, e.g., transitions $t_1^{r_{11}}$, $t_1^{r_{12}}$ and $t_1^{r_{13}}$ are co-located.

Co-locating transitions leads to one more way of enabling for steps of transitions. We say that a *step* U of *PTL* is *lmax-enabled* at a marking M if $M[U\rangle_{free}$ and U cannot be extended by a transition co-located with a transition in U to yield a step which is free-enabled at M; i.e., there is no $t \in T$ such that $\ell(t) \in \ell(U)$ and $M[U + \{t\}\rangle_{free}$. We denote this by $M[U\rangle_{lmax}$, and then denote the lmax-execution of U by $M[U\rangle_{lmax}M'$, where $M' = M - {}^\bullet U + U^\bullet$. Note that *locally maximal (lmax) concurrency* is similar to maximal concurrency, but now only active localities[1] cannot execute further transitions. For the PTL-net in Figure 2 we have that $\{t_1^{r_{11}}, t_1^{r_{12}}\}$ is lmax-enabled at the given marking, but $\{t_1^{r_{11}}\}$ is not.

Let $\mathfrak{m} \in \{free, max, lmax\}$ be a mode of execution of a PTL-net *PTL*. Then an $\mathfrak{m}$-*step sequence* is a finite sequence of $\mathfrak{m}$-executions starting from the initial marking, and an $\mathfrak{m}$-*reachable* marking is any marking resulting from the execution of such a sequence. Moreover, the $\mathfrak{m}$-*concurrent reachability graph of PTL* is the step transition system:

$$CRG_{\mathfrak{m}}(PTL) = \Big([M_0\rangle_{\mathfrak{m}} \ , \ \big\{(M, U, M') \mid M \in [M_0\rangle_{\mathfrak{m}} \wedge M[U\rangle_{\mathfrak{m}} M'\big\} \ , \ M_0 \Big) \ ,$$

where $[M_0\rangle_{\mathfrak{m}}$ is the set of all $\mathfrak{m}$-reachable markings which are the nodes of the graph; M_0 is the initial node; and the arcs between the nodes are labelled by $\mathfrak{m}$-executed steps of transitions. Concurrent reachability graphs provide complete representations of the dynamic behaviour of PTL-nets evolving according to the chosen mode of execution.

[1] By active localities of a step U we mean the localities of transitions present in U.

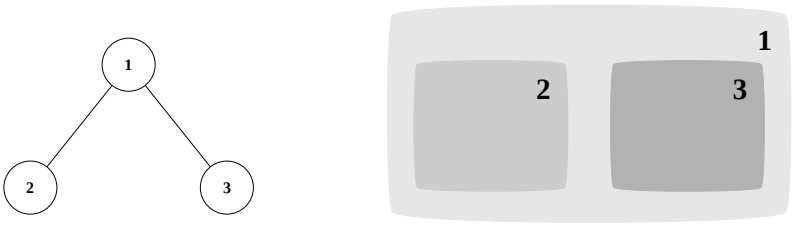

Figure 3: A membrane structure ($m = 3$) and its compartments with 1 being the root node, $(1,2) \in \mu$ and $1 = \mathit{parent}(3)$.

Membrane structures. 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 indicate that there is an edge from i (parent) to j (child) in the tree of μ, and $i \in \mu$ means that i is a node of μ. The nodes of a membrane structure represent nested membranes which in turn determine compartments (compartment i is enclosed by membrane i and lies in-between i and its children, if any), as shown in Figure 3.

We will say that a PTL-net $PTL = (P,T,W,\ell,M_0)$ is *spanned* over the membrane structure μ if $\ell : P \cup T \to \mu$ and the following hold, for all $p \in P$ and $t \in T$:

- if $W(p,t) > 0$ then $\ell(p) = \ell(t)$; and
- if $W(t,p) > 0$ then $\ell(p) = \ell(t)$ or $(\ell(p),\ell(t)) \in \mu$ or $(\ell(t),\ell(p)) \in \mu$.

The PTL-net of Figure 2 is spanned over the membrane structure depicted in Figure 3.

Basic membrane systems. Let V be a finite alphabet of names of *objects* (or molecules) and let μ be a membrane structure of degree m. A *basic membrane system* (over V and μ) is a tuple

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

such that, for every membrane i, w_i^0 is a multiset of objects from V, and R_i is a finite set of *evolution rules* associated with membrane (compartment) i. Each evolution rule $r \in R_i$ is of the form $r : lhs^r \to rhs^r$, where lhs^r (the left hand side of r) is a nonempty multiset over V, and rhs^r (the right hand side of r) is a multiset over

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

Here 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 *BMS* 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 4 shows a basic membrane system over the membrane structure depicted in Figure 3.

A membrane system evolves from configuration to configuration as a consequence of the application of evolution rules. There are different *execution modes* ranging from fully synchronous — as many applications of rules as possible — to sequential — a single application of a rule at a time. Here, similarly as in the case of PTL-nets, we distinguish three modes, all based on the notion of a vector multi-rule.

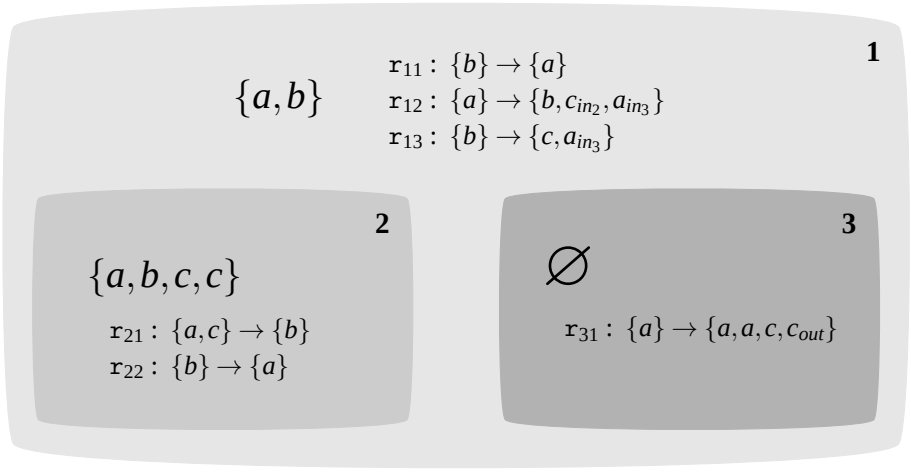

Figure 4: Basic membrane system BMS_0.

A *vector multi-rule* of *BMS* 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 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 of evolution rules in $\mathbf{r}$.

A vector multi-rule $\mathbf{r}$ of *BMS* is

- *free-enabled* at a configuration C if $lhs_i^{\mathbf{r}} \leq w_i$, for each i.

Moreover, a free-enabled vector multi-rule $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$ is:

- *max-enabled* if no $\mathbf{r}_i$ can be extended to a vector multi-rule which is free-enabled at C; and
- *lmax-enabled* if no nonempty $\mathbf{r}_i$ can be extended to a vector multi-rule which is free-enabled at C.

For example, in Figure 4,

- $\langle \varnothing, \varnothing, \{\mathtt{r}_{31}\} \rangle$ is not free-enabled;
- $\langle \{\mathtt{r}_{11}, \mathtt{r}_{12}\}, \varnothing, \varnothing \rangle$ is lmax-enabled but not max-enabled; and
- $\langle \{\mathtt{r}_{11}, \mathtt{r}_{12}\}, \{\mathtt{r}_{21}, \mathtt{r}_{22}\}, \varnothing \rangle$ 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 any more rules; in other words, each compartment either uses no rule, or uses a maximal multiset of rules.

The effect of the rules is independent of the mode of execution $\mathfrak{m} \in \{free, max, lmax\}$. 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 finite sequence of $\mathfrak{m}$-evolutions starting from the initial configuration; any configuration which can be obtained through such a computation is called $\mathfrak{m}$-*reachable*. For the basic membrane system depicted in Figure 4 we have, for example:

$$C_0 \xrightarrow{\langle\{\mathbf{r}_{11},\mathbf{r}_{12}\},\varnothing,\varnothing\rangle}_{lmax} (\{a,b\},\{a,b,c,c,c\},\{a\}) \xrightarrow{\langle\varnothing,\{\mathbf{r}_{21},\mathbf{r}_{22}\},\varnothing\rangle}_{lmax} (\{a,b\},\{a,b,c,c\},\{a\}) \ .$$

Let $\mathfrak{m} \in \{free, max, lmax\}$ be a mode of execution of a basic membrane system *BMS*. Then the $\mathfrak{m}$-*concurrent reachability graph of BMS* is given by:

$$CRG_{\mathfrak{m}}(BMS) = \Big([C_0\rangle_{\mathfrak{m}} \ , \ \big\{(C, \mathbf{r}_1 + \ldots + \mathbf{r}_m, C') \mid C \in [C_0\rangle_{\mathfrak{m}} \wedge C \xrightarrow{\langle \mathbf{r}_1,\ldots,\mathbf{r}_m\rangle}_{\mathfrak{m}} C'\big\} \ , \ C_0 \Big) \ ,$$

where $[C_0\rangle_{\mathfrak{m}}$ is the set of all $\mathfrak{m}$-reachable configurations which are the nodes of the graph; C_0 is the initial node; and the arcs between the nodes are labelled by multisets of evolution rules involved in the $\mathfrak{m}$-executed vector multi-rules.[2] Similarly as in the case of PTL-nets, concurrent reachability graphs capture completely the dynamic behaviour of basic membrane system evolving according to the chosen mode of execution.

3 Membrane Systems and Petri Nets

There is a natural way of translating a basic membrane system $BMS = (V, \mu, w^0_1, \ldots, w^0_m, R_1, \ldots, R_m)$ over a membrane structure μ into a behaviourally equivalent PTL-net $PTL(BMS) = (P, T, W, \ell, M_0)$ spanned over the same membrane structure. In the constructed net, places represent objects present inside compartments, and transitions represent evolution rules. Both places and transitions are associated with membranes and this information is represented by the location mapping.

The constructed PTL-net $PTL(BMS)$ has a separate place $\mathtt{p}^a_j$ with $\ell(\mathtt{p}^a_j) = j$, for each object a and membrane j, and a separate transition $\mathtt{t}^r_i$ with $\ell(\mathtt{t}^r_i) = i$, for each rule r in compartment i.

The initial marking inserts $w^0_j(a)$ tokens into each place $\mathtt{p}^a_j$. The connectivity between transition $t = \mathtt{t}^r_i$ and place $p = \mathtt{p}^a_j$ is given by:

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

as well as:

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

[2] Though it may be that rules from different membranes are the same in terms of the multisets defining their left hand and right hand sides, we assume here that evolution rules associated with different membranes can be distinguished, e.g., by giving them each their own name (an injective label).

Figure 2 shows the result of the above translation for the basic membrane system in Figure 4. Note that it immediately follows from the construction that the PTL-net $PTL(BMS)$ is spanned over μ.

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

- for every marking M of $PTL(BMS)$, $\nu(M) = (w_1, \ldots, w_m)$ is the configuration of *BMS* given by $w_i(a) = M(\mathtt{p}_i^a)$, for every object a and every i.
- for every step U of $PTL(BMS)$, $\rho(U) = \langle \mathbf{r}_1, \ldots, \mathbf{r}_m \rangle$ is the vector multi-rule of *BMS* given by $\mathbf{r}_i(r) = U(\mathtt{t}_i^r)$, for every rule $r \in R_i$ and every 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 $\mathfrak{m} \in \{free, max, lmax\}$, configurations C of *BMS* and markings M of $PTL(BMS)$. Together with $\nu(M_0) = C_0$, this result means that the $\mathfrak{m}$-step sequences of $PTL(BMS)$ faithfully represent $\mathfrak{m}$-computations of *BMS*, and the same applies to markings and configurations. Crucially, we obtain

Theorem 1 *For each* $\mathfrak{m} \in \{free, max, lmax\}$,

$$CRG_{\mathfrak{m}}(PTL(BMS)) \sim_{\phi,\nu} CRG_{\mathfrak{m}}(BMS)\ ,$$

where the mapping ν *is defined as above, and* $\phi(\mathtt{t}_i^r) = r$, *for every transition* $\mathtt{t}_i^r$ *of* $PTL(BMS)$.

The above theorem captures the very tight behavioural correspondence between *BMS* and $PTL(BMS)$, allowing to apply analytical techniques developed for Petri nets in the analysis of membrane systems. For example, one can employ the invariant analysis based on linear algebra [26], or use the causality semantics approach of Petri nets based on occurrence nets, as first outlined in [12]. In this paper, we show how techniques used to synthesise Petri nets could be employed in order to construct basic membrane systems from their intended behaviours as represented by step transition systems. First, however, we provide a translation from PTL-nets spanned over membrane structures to basic membrane systems.

Let $PTL = (P, T, W, \ell, M_0)$ be a PTL-net spanned over a membrane structure μ. For such a PTL-net, we construct the corresponding basic membrane system $BMS(PTL)$ over μ in the following way:

- P is the set of objects;
- the initial configuration is $\nu'(M_0)$ where, for every marking M of PTL,

$$\nu'(M) = (M|_{\ell^{-1}(1)\cap P}, \ldots, M|_{\ell^{-1}(m)\cap P})\ ;$$

- each transition $t \in T$ with $t^\bullet = \{p^1, \ldots, p^k\}$ has a corresponding evolution rule $\phi'(t)$ of the form $t : {}^\bullet t \to \{a_1, \ldots, a_k\}$ where, for $i = 1, \ldots, k$,

$$a_i = \begin{cases} p^i & \text{if } \ell(p^i) = \ell(t) \\ p^i_{out} & \text{if } \ell(p^i) = parent(\ell(t)) \\ p^i_{in_{\ell(p^i)}} & \text{if } \ell(t) = parent(\ell(p^i)) \end{cases}$$

- for each membrane $i \in \mu$, the set of evolution rules is given by $R_i = \{\phi'(t) \mid \ell(t) = i\}$.

Again, the translation results in a very close behavioural correspondence.

Theorem 2 *For each* $\mathfrak{m} \in \{free, max, lmax\}$,

$$CRG_{\mathfrak{m}}(PTL) \sim_{\phi',\nu'} CRG_{\mathfrak{m}}(BMS(PTL)) \ ,$$

where the mappings ϕ' *and* ν' *are defined as above.*

It follows from Theorems 1 and 2 that the problem of synthesis of basic membrane systems from step transition systems is equivalent to the problem of synthesis of PTL-nets spanned over membrane structures. It therefore suffices to solve the latter, and in the next section we describe a solution based on the notion of a region of a step transition system.

4 Synthesising nets corresponding to membrane systems

The Petri net synthesis problem we consider is formulated as follows.

Problem 1 *Given are a finite set* T*, a membrane structure* μ*, a mapping* $\ell : T \to \mu$*,* $\mathfrak{m} \in \{free, max, lmax\}$*, and* $TS = (Q, \mathscr{A}, q_0)$ *which is a finite step transition system over* T*.*
Construct a PTL*-net* $PTL = (P, T, \ell, M_0)$ *spanned over* μ *such that* $CRG_{\mathfrak{m}}(PTL) \sim TS$*, and* ℓ *is an extension of the mapping defined for* T*.*

As demonstrated in [4], synthesis problems like Problem 1 can be solved using techniques coming from the theory of regions of transition systems (see, e.g., [1, 7, 18]). Intuitively, a region represents a single place in a hypothetical net generating the given transition system. Regions are used both to check whether a net satisfying the conditions can be constructed and, if the answer turns out to be positive, to construct such net.

In this particular case, a *region* of the step transition system *TS* consists of three mappings

$$reg = \big(\ \sigma : Q \to \mathbb{N} \ , \ \imath : T \to \mathbb{N} \ , \ \omega : T \to \mathbb{N} \ \big) \tag{2}$$

such that, for every arc $q \xrightarrow{\alpha} q'$ of *TS*,

$$\sigma(q) \geq \omega(\alpha) \quad \text{and} \quad \sigma(q') = \sigma(q) - \omega(\alpha) + \imath(\alpha) \ . \tag{3}$$

Here $\omega(\alpha) = \sum_{t \in T} \alpha(t) \cdot \omega(t)$ and similarly $\imath(\alpha) = \sum_{t \in T} \alpha(t) \cdot \imath(t)$. In a region of the form (2) representing a place p, $\sigma(q)$ is the number of tokens in p in the marking corresponding to the node q, $\omega(t)$ represents the weight of the arc from p to transition t, and $\imath(t)$ represents the weight of the arc from t to p. It is then natural to require in (3) that p contains enough tokens not to block a step α executed at q, and also to ensure that the number of tokens in p before and after executing α is consistent with the total arc weight of the step α in relation to p.

In the case of Problem 1, one also needs to take into account the fact that the target PTL-net must be spanned over μ. This imposes additional constraints on allowed regions (places) and the location mapping ℓ. We call a region *reg* as in (2) with a location $\ell(reg) \in \mu$ *compatible with* the membrane structure μ if the following hold, for every $t \in T$:

- if $\omega(t) > 0$ then $\ell(t) = \ell(reg)$; and
- if $\imath(t) > 0$ then $\ell(t) = \ell(reg)$ or $(\ell(t), \ell(reg)) \in \mu$ or $(\ell(reg), \ell(t)) \in \mu$.

The set of all such regions will be denoted by $\mathbf{P}_\mu$. Note that if *reg* is such that $\omega(t) > 0$, for at least one $t \in T$, then $\ell(reg)$ is uniquely determined; otherwise we always choose $\ell(reg)$ to be the membrane which is higher up in the tree structure of μ than any other suitable candidate. As a result, we can leave $\ell(reg)$ implicit.

Finally, Problem 1 should be feasible in the sense that the transition system *TS* can be realised by a suitable net. There are two necessary and sufficient conditions for realisability (see [4, 17]):

- *state separation*: for every pair of distinct states of the transition system there is a region (a marked place) distinguishing between them; and
- *forward closure*: there are sufficiently many places defined by regions of the transition system to disallow steps not present in the transition system.

First we describe how all places can be found that potentially provide a solution to Problem 1; in other words, all the regions (2) of the transition system *TS* which are compatible with μ.

Finding compatible regions. Let T, μ, $\ell : T \to \mu$ and $TS = (Q, \mathscr{A}, q_0)$ be as in Problem 1. Assume that $Q = \{q_0, \ldots, q_h\}$ and $T = \{t_1, \ldots, t_n\}$. We use three vectors of non-negative variables:

$$\mathbf{x} = x_0 \ldots x_h \qquad \mathbf{y} = y_1 \ldots y_n \qquad \mathbf{z} = z_1 \ldots z_n \ .$$

We also denote $\mathbf{p} = \mathbf{xyz}$ and define a homogeneous linear system

$$\mathscr{P} \ : \ \begin{cases} x_i \geq \alpha \cdot \mathbf{z} \\ x_j = x_i + \alpha \cdot (\mathbf{y} - \mathbf{z}) \end{cases} \qquad \text{for all } q_i \xrightarrow{\alpha} q_j \text{ in } TS$$

where $\alpha \cdot \mathbf{z}$ denotes $\alpha(t_1) \cdot z_1 + \cdots + \alpha(t_n) \cdot z_n$ and similarly for $\alpha \cdot (\mathbf{y} - \mathbf{z})$.

The regions (2) of *TS* are then determined by the integer solutions $\mathbf{p}$ of the system $\mathscr{P}$ assuming that, for $0 \leq i \leq h$ and $1 \leq j \leq n$,

$$\sigma(q_i) = x_i \qquad \iota(t_j) = y_j \qquad \omega(t_j) = z_j$$

The set of rational solutions of $\mathscr{P}$ forms a polyhedral cone in $\mathbb{Q}^{h+2n+1}$. As described in [3], one can effectively compute finitely many integer generating rays $\mathbf{p}^1, \ldots, \mathbf{p}^k$ of this cone such that any integer solution $\mathbf{p}$ of $\mathscr{P}$ can be expressed as a linear combination of the rays with non-negative rational coefficients:

$$\mathbf{p} = \sum_{l=1}^{k} c_l \cdot \mathbf{p}^l \ .$$

Such rays $\mathbf{p}^l$ are fixed and (some of them) turned into net places if Problem 1 has a solution. More precisely, if $\mathbf{p}^l$ is included in the constructed net, then

$$M_0(\mathbf{p}^l) = x_0^l \qquad W(\mathbf{p}^l, t_i) = z_i^l \qquad W(t_i, \mathbf{p}^l) = y_i^l \ , \tag{4}$$

where M_0 is the initial marking of the target net, and $t_i \in T$.

Clearly, not all such rays can be considered for the inclusion in the net being constructed, as the corresponding regions have to be compatible with μ. We therefore ensure through a simple check that the generating rays $\mathbf{p}^1, \ldots, \mathbf{p}^k$ are compatible with μ, deleting in the process those which are not. Note that any $\mathbf{p} \in \mathbf{P}_\mu$ is a non-negative linear combination of rays compatible with μ.

Having found the generating rays compatible with μ, we proceed to check whether Problem 1 has any solutions at all.

Checking state separation. Let $TS = (Q, \mathscr{A}, q_0)$ be as in Problem 1. We take in turn each pair of distinct states, q_i and q_j, of Q and decide whether there exists $\mathbf{p} = (\sigma, \iota, \omega) \in \mathbf{P}_\mu$ with coefficients $c_1, \ldots, c_k$ such that $\sigma(q_i) = x_i \neq x_j = \sigma(q_j)$. Since the latter is equivalent to

$$\sum_{l=1}^{k} c_l \cdot x_i^l \neq \sum_{l=1}^{k} c_l \cdot x_j^l \,,$$

one can simply check whether there exists at least one $\mathbf{p}^l$ (called a *witness* [6]) such that $x_i^l \neq x_j^l$.

Checking forward closure. Again, let $TS = (Q, \mathscr{A}, q_0)$ be as in Problem 1, and $\mathfrak{m} \in \{free, max, lmax\}$. First, we take in turn each state q_i of Q, and calculate the set of *region enabled* steps, denoted by $regSteps_{q_i}$. Intuitively, region enabled steps are those that cannot be disabled (or blocked) by compatible regions.

To build $regSteps_{q_i}$ one only needs to consider nonempty steps α with $|\alpha| \leq m \cdot Max$, where Max is the maximum size of steps labelling arcs in TS, and m is the number of membranes of μ. The reason is that, for each membrane $i \in \mu$ there exists a compatible region $(\sigma, \iota, \omega) \in \mathbf{P}_\mu$ (called a *witness*) such that $\sigma(Q) = \{Max\}$ and, for every $t \in T$,

$$\omega(t) = \iota(t) = \begin{cases} 1 & \text{if } \ell(t) = i \\ 0 & \text{otherwise} \,. \end{cases}$$

Taken together, all such regions block any step α with $|\alpha| > m \cdot Max$.

For each nonempty step α with $|\alpha| \leq m \cdot Max$ it is the case that $\alpha \notin regSteps_{q_i}$ *iff* for some $\mathbf{p} \in \mathbf{P}_\mu$ with coefficients $c_1, \ldots, c_k$ we have $x_i < \alpha \cdot \mathbf{z}$. Since the latter is equivalent to

$$\sum_{l=1}^{k} c_l \cdot (x_i^l - \alpha \cdot \mathbf{z}^l) < 0 \,,$$

one simply checks whether there exists at least one $\mathbf{p}^l$ (again called a *witness*) such that $x_i^l - \alpha \cdot \mathbf{z}^l < 0$.

Having determined the region enabled steps, in order to establish forward closure we need to verify that, for every state $q \in Q$,

$$tsSteps_q = \begin{cases} regSteps_q & \text{if } \mathfrak{m} = free \\ \{\alpha \in regSteps_q \mid \neg\exists t \in T : \ \alpha + \{t\} \in regSteps_q\} & \text{if } \mathfrak{m} = max \\ \{\alpha \in regSteps_q \mid \neg\exists t \in T : \ \alpha + \{t\} \in regSteps_q \wedge \ell(t) \in \ell(\alpha)\} & \text{if } \mathfrak{m} = lmax \,. \end{cases}$$

Constructing the solution net. If the above checks for the feasibility of Problem 1 are successful, one can construct a solution PTL-net spanned over μ by taking all the witness rays and regions, and treating them as places in the way indicated in (4). The resulting net PTL satisfies

$$CRG_{\mathfrak{m}}(PTL) \sim TS \,.$$

5 Concluding remarks

We have described how one can adapt a solution to the Petri net synthesis problem based on regions of step transition systems, so that the resulting method can be used to construct basic membrane systems

with a specific behaviour. Moreover, there are other synthesis results developed for Petri nets which can be employed to extend the proposed solution in several directions, two of which are briefly mentioned below.

In Problem 1 it is assumed that the association of transitions with membranes is given. This can be relaxed and one can aim at synthesising membrane systems without such an association, or even without being given a membrane structure (in such a case, the synthesis procedure should construct a membrane structure as well). For such a modification, there already exist results which can be used to develop a solution. More precisely, the method of 'discovering' localities in [17] works for PTL-nets with localised conflicts (where transitions which share an input place are co-located). Since all PTL-nets spanned over membrane structures have localised conflicts, the result in [17] can be adapted to work for basic membrane systems.

Evolution rules of membrane systems are often equipped with promoters and inhibitors. Both features have direct counterparts in Petri nets in the form of activator and inhibitor arcs, and suitable translations between membrane systems and Petri nets can be developed as described in [9, 14]. Moreover, the synthesis technique based on regions of step transition systems works also for PTL-nets extended with activator and inhibitor arcs [16]. In fact, there is a general setting of so-called τ-nets and corresponding τ-regions [1, 4]. Here the parameter τ is a general and convenient way of capturing different types of connections (arcs and their combinations) between places and transitions, removing the need to re-state and re-prove the key results every time a new kind arcs is introduced. Note that the recently introduced SET-nets [13, 14] (with qualitative rather than quantitative resource management) and set membrane systems [10] can be treated within the general theory of τ-net synthesis based on regions of transition systems [11].

Acknowledgements

This paper is based on an invited talk presented at the 6th Workshop on Membrane Computing and Biologically Inspired Process Calculi (MECBIC), 8th September 2012, Newcastle upon Tyne, United Kingdom. The reported research was supported by the EPSRC GAELS project.

References

[1] E.Badouel and Ph.Darondeau: Theory of Regions. In: Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models, Advances in Petri Nets. Lecture Notes in Computer Science **1491**. Springer-Verlag, Berlin Heidelberg New York (1998) 529–586, doi:10.1007/3-540-65306-6_22

[2] L.Bernardinello: Synthesis of Net Systems In: Marsan, M.A. (ed.): Application and Theory of Petri Nets 1993. Lecture Notes in Computer Science **691**. Springer-Verlag, Berlin Heidelberg New York (1993) 89–105, doi:10.1007/3-540-56863-8_42

[3] N.Chernikova: Algorithm for Finding a General Formula for the Non-negative Solutions of a System of Linear Inequalities. USSR Computational Mathematics and Mathematical Physics **5** (1965) 228–233

[4] P.Darondeau, M.Koutny, M.Pietkiewicz-Koutny and A.Yakovlev: Synthesis of Nets with Step Firing Policies. Fundamenta Informaticae **94** (2009) 275–303

[5] J.Desel and G.Juhas: What Is a Petri Net? Lecture Notes in Computer Science **2128**, Springer-Verlag (2001) 1–25, doi:10.1007/3-540-45541-8_1

[6] J.Desel and W.Reisig: The Synthesis Problem of Petri Nets. Acta Informatica **33** (1996) 297–315, doi:10.1007/s002360050046

[7] A.Ehrenfeucht and G.Rozenberg: Partial 2-structures; Part I: Basic Notions and the Representation Problem, and Part II: State Spaces of Concurrent Systems. Acta Informatica **27** (1990) 315–368, doi:10.1007/BF00264611, doi:10.1007/BF00264612

[8] J.Kleijn and M.Koutny: Processes of Membrane Systems with Promoters and Inhibitors. Theoretical Computer Science **404** (2008) 112–126, doi:10.1016/j.tcs.2008.04.006

[9] J.Kleijn and M.Koutny: Petri Nets and Membrane Computing. In: [22] (2010) 389–412

[10] J.Kleijn and M.Koutny: Membrane Systems with Qualitative Evolution Rules. Fundamenta Informaticae **110** (2011) 217–230, doi:10.3233/FI-2011-539

[11] J.Kleijn, M.Koutny, M.Pietkiewicz-Koutny and G.Rozenberg: Step Semantics of Boolean Nets. Acta Informatica (2012), doi:10.1007/s00236-012-0170-2

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

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

[14] J.Kleijn, M.Koutny and G.Rozenberg: Petri Nets for Biologically Motivated Computing. Scientific Annals of Computer Science **21** (2011) 199–225

[15] I.Koch, W.Reisig and F.Schreiber: Modeling in Systems Biology — The Petri Net Approach. Springer Verlag (2010)

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

[17] M.Koutny and M.Pietkiewicz-Koutny: Synthesis of Petri Nets with Localities. Scientific Annals of Computer Science **19** (2009) 1–23

[18] M.Mukund: Petri Nets and Step Transition Systems. International Journal of Foundations of Computer Science **3** (1992) 443–478, doi:10.1142/S0129054192000231

[19] G.Păun: Computing with Membranes. J. Comput. Syst. Sci. **61** (2000) 108–143, doi:10.1006/jcss.1999.1693

[20] G.Păun: Membrane Computing, An Introduction. Springer-Verlag, Berlin Heidelberg New York (2002)

[21] G.Păun and G.Rozenberg: A Guide to Membrane Computing. Theoretical Computer Science **287** (2002) 73–100, doi:10.1016/S0304-3975(02)00136-6

[22] G.Păun, G.Rozenberg and A.Salomaa: The Oxford Handbook of Membrane Computing. Oxford University Press (2010)

[23] C.A.Petri: Kommunikation mit Automaten. PhD Thesis (1962)

[24] M.Pietkiewicz-Koutny: The Synthesis Problem for Elementary Net Systems with Inhibitor Arcs. Fundamenta Informaticae **40** (1999) 251–283

[25] W.Reisig and G.Rozenberg (eds.): Lectures on Petri Nets. Lecture Notes in Computer Science **1491,1492** (1998), doi:10.1007/3-540-65306-6_19

[26] M.Silva, E.Teruel and J.M.Colom: Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems. Lecture Notes in Computer Science, Springer-Verlag **1491** (1998) 309–373, doi:10.1007/3-540-65306-6_19