Journal of Membrane Computing (2020) 2:341–354
https://doi.org/10.1007/s41965-020-00062-y

REGULAR PAPER

Description of membrane systems with time Petri nets: promoters/inhibitors, membrane dissolution, and priorities

Péter Battyányi[1] · **György Vaszil**[1]

Received: 21 June 2020 / Accepted: 9 October 2020 / Published online: 27 October 2020

Abstract
We continue the investigations of the connection between membrane systems and time Petri nets by extending the examined class of systems from simple symbol-object membrane systems to more complex cases: rules with promoters/inhibitors, membrane dissolution, and priority relation on the rules. By constructing the simulating time Petri net, we retain one of the main characteristics of the Petri net model; namely, the firings of the transitions can take place in any order, and there is no need to introduce maximal parallelism in the Petri net semantics. Instead, we substantially exploit the gain in computational strength obtained by the introduction of the timing feature for Petri nets.

Keywords Petri nets · Promoters and inhibitors · Priorities · Membrane dissolution

1 Introduction

Several models have emerged in the past decades to model distributed systems with interactive, parallel components. One of them was developed by Petri [16], and since then, the Petri nets have become the underlying system of a vast field of research with a considerable practical interest, see [3, 15, 19] for more information. The theory of membrane systems was established by Păun [12], and it has proved to be a very convenient and many-sided model of distributed systems with concurrent processes, see [13, 14]. Here, we continue the investigations concerning the relationship of these two computational models.

Place/transition Petri nets are bipartite graphs, the conditions of the events of a distributed system are represented by places, and directed arcs connect the places to the transitions which model the events. The conditions for the events are expressed by tokens: an event can take place, i.e., a transition can fire, if there are enough tokens in the places at the source ends of the incoming arcs of a transition. These places are called preconditions. The outgoing edges of a transition represent the post-condition of the events. Firing of a transition means removing tokens from the preconditions and adding them to the post-conditions. The number of tokens moved in this way is prescribed by the multiplicities of the incoming and outgoing arcs.

Membrane systems are models of distributed, synchronized computational systems ordered in a tree-like structure. The building blocks are compartments which contain multisets of objects. The multisets evolve in each compartment in a parallel manner, and the compartments, in each computational step, wait for the others to finish their computation; hence, the system acts in a synchronized manner. In every computational step, the multisets in the compartments evolve in a maximal parallel manner. This means that as many evolution rules of the compartment are applied simultaneously in each step as possible. For more on the ways of synchronizing P systems, see the handbook [14] and the recent papers [2, 4].

By looking at the basic functioning of membrane systems and place/transition nets, we might notice some similar features. Petri net transitions consume tokens from their input places and produce new tokens at their output places, so in some sense they behave similarly to membrane systems which consume, produce, and move objects around in the

The preliminary version of this paper was presented at CMC20, the 20th Conference on Membrane Computing, August 5–8, 2019, in Curtea de Argeş, Romania.

✉ György Vaszil
vaszil.gyorgy@inf.unideb.hu

Péter Battyányi
battyanyi.peter@inf.unideb.hu

[1] Department of Computer Science, Faculty of Informatics, University of Debrecen, Kassai út 26, Debrecen 4028, Hungary

regions of their membrane structure. And not only do they behave similarly, but the functioning of place/transition nets can naturally be described by transformations of the multisets corresponding to possible token distributions on the places of the net. See [5, 9, 10] for more on these and similar ideas. As we will describe later in more detail, each kind of object in a compartment of a membrane system can be represented by a different place, and each evolution rule by a different transition having its input and output places. The building of such a structural link between the two models motivates the study of membrane systems from the point of view of the concurrent nature of their behavior, and as a consequence, the techniques and tools developed for Petri nets might become applicable in the area of membrane computing.

Besides this basic connection described above, Petri net variants were also introduced with the aim of capturing some of the advance features of more sophisticated membrane system models. For example, in order to describe the maximal parallel application of rules in membrane systems, the usual semantics of place/transition nets can be extended. So-called maximally concurrent steps in Petri nets were already considered in [6] (independently of any motivation from membrane computing) to describe the concurrent evolution of non-sequential systems. In order to capture the compartmental structure and the locally maximal parallel behavior of membrane systems, localities and a locally maximal concurrent semantics were introduced and investigated for Petri nets in [9]. Furthermore, Petri nets with localities were also used to describe membrane creation and membrane dissolution in [8] and extended in [7] to model the use of promoters and inhibitors. For more details on the specific membrane system models and their computational power, see the handbook [14, 20, 21] for more recent developments. On the connections of these models to variants of Petri nets with localities, the reader is also referred to [14].

In the present paper, we follow a different approach by considering the possibilities provided by a model called time Petri net developed by Merlin [11] in order to deal with the difficulty that ordinary place/transition nets are not able to model systems where a certain order of events must be taken into account, see also [17, 18]. In this model, time intervals are associated with transitions. The local time observed from a transition can be modified by the Petri net state transition rules, and a transition can fire only if its observed time lies in the interval assigned to the transition by the definition of the net. In this way, the computational power of Petri nets is increased: the time Petri net model is Turing complete in contrast with the original state/transition Petri net.

Besides the increase of the computational power, the feature of timing also provides a more or less natural framework to describe the computations of membrane systems more conveniently. In the following, we continue the research on the connection between time Petri nets and membrane systems initiated in [1]. One of the main features of our construction is that the usual semantic characteristics of Petri nets is retained when a Petri net equivalent of a membrane system is presented. That is, unlike the construction in [9] (and unlike the constructions in many other Petri net descriptions of membrane systems), we do not stipulate that the Petri nets should perform their computational steps in some variant of a maximal concurrent manner. Instead of a modified membrane system-like semantics, the attached time intervals provide the synchronization in the corresponding Petri nets.

The structure of the paper is the following. After providing the necessary preliminaries and definitions in Sect. 2, we describe in Sect. 3 a variant of the basic construction of the time Petri net simulation of symbol object membrane systems developed in [1]. Then, in Sect. 4, this construction is extended in order to represent some more advanced membrane computational tools. Similarly to [1], there will be no need to introduce any other "special features" (beside the feature of time) to be able to capture the effect of priorities, the use of promoters/inhibitors, or membrane dissolution. Finally, the paper ends with a short section of conclusions.

2 Preliminaries and definitions

A finite multiset over an alphabet O is a mapping $M : O \to \mathbb{N}$, where $\mathbb{N}$ is the set of nonnegative integers. The number $M(a)$ for $a \in O$ is called the multiplicity of a in M. We write that $M_1 \subseteq M_2$ if for all $a \in O$, $M_1(a) \leq M_2(a)$. The union or sum of two multisets over O is defined as $(M_1 + M_2)(a) = M_1(a) + M_2(a)$, the intersection is $(M_1 \cap M_2)(a) = \min(M_1(a), M_2(a))$, while the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2)(a) = M_1(a) - M_2(a)$ for all $a \in O$. The set of all finite multisets over an alphabet O is denoted by $\mathcal{M}(O)$; the empty multiset is denoted by $\emptyset$.

The notation $\mathbb{N}_{>0}$ stands for the set of positive integers, while $\mathbb{Q}$ and $\mathbb{Q}_{\geq 0}$ denote the set of rational numbers and nonnegative rational numbers and $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ the set of real numbers and nonnegative real numbers, respectively.

2.1 Membrane systems

Membrane systems are computational models operating on multisets. We define the notion of the basic symbol-object membrane system [14] and shortly introduce the additional features which will be discussed in more detail in Sect. 5. A membrane system (or P system) is a tree-like structure of hierarchically arranged membranes. The outermost membrane is usually called the skin membrane. The membranes are labeled by natural numbers $\{1, \ldots, n\}$, and each membrane, except the skin membrane, has its parent

membrane. We use μ for representing the structure of the membrane system itself, which can also be given as a balanced string of left and right brackets indexed by their labels. If $\mu = [_1\,[_2\,[_3]_3\,[_4]_4\,]_2\,[_5]_5\,]_1$, then the skin membrane has two submembranes, while region also contains two embedded regions. Abusing the notation, $\mu(i) = k$ can also mean that the parent of the i-th region is region k.

The regions of a P system contain multisets over a finite alphabet O of objects. The contents of the regions evolve through rules associated with the regions. The rules describe how certain multisets can be transformed to other multisets. They are applied in a maximal parallel manner, which constitute the "micro-steps" of the computations. A computational step is the "macro-step" of the process: it ends when each of the regions has finished the parallel applications of their rules. A computational sequence is a sequence of computational steps.

Here, we think of the computational steps in the regions as consisting of two phases: first the rule application phase produces from the objects on the left-hand sides of the rules the labeled objects on the right-hand sides. (The labels of the labeled objects describe the way they should be moved between the regions: stay where they are, move to the parent region, or move into one of the child regions.) Then, we have the communication phase when the labels (also called target indicators) are removed and all the objects are transported to the regions indicated by their labels. The P system gives a result when it halts, i.e., when no more rules can be applied in any of the regions. The result is a natural number or a tuple of natural numbers counting certain objects in the membrane(s) designated as the output membrane(s).

A P system of degree $n \geq 1$ is $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$ where O is an alphabet of objects, μ is a membrane structure of n membranes, $w_i \in \mathcal{M}(O)$ with $1 \leq i \leq n$ are the initial contents of the n regions, R_i with $1 \leq i \leq n$ are the sets of evolution rules associated with the regions. They are of the form $u \to v$, where $u \in \mathcal{M}(O)$ and $v \in \mathcal{M}(O \times \text{tar})$ with $\text{tar} = \{\text{here}, \text{out}\} \cup \{\text{in}_j \mid 1 \leq j \leq n\}$.

We assume that one or more membranes are designated as output membranes. A configuration is a sequence $W = (u_1, \ldots, u_n)$, where u_i is the multiset contained by membrane $i, 1 \leq i \leq n$. For a rule $r : u \to v \in R_i$, we denote by $\text{lhs}(r) \in \mathcal{M}(O)$ and $\text{rhs}(r) \in \mathcal{M}(O \times \text{tar})$ the left-hand side and the right-hand side of r, respectively (u and v, for the rule $u \to v$). By the application of a rule $u \to v \in R_i$, we mean the process of removing the elements of u from the multiset u_i and extending u_i with the labeled elements, which are called messages. As a result, during a computational step, a region can contain both elements of O and messages from $O \times \text{tar}$. We say that W is a proper configuration if $u_i \in \mathcal{M}(O)$ for each $1 \leq i \leq n$, while W is an intermediate configuration if there is at least one region containing elements from $O \times \text{tar}$, that is, $u_i \cap (Q \times \text{tar}) \neq \emptyset$ for some $i,\ 1 \leq i \leq n$.

The communication phase after the rule applications produces proper configurations from intermediate ones: the elements coming from the right-hand sides of the rules are added to the regions specified by the target indicators associated with them. If the right-hand side of a rule contains a pair $(a, \text{here}) \in O \times \text{tar}$, then a is added to the region where the rule is applied. If it contains (a, out), then a is added to the parent region; if it contains (a, in_j), then a is added to the contents of region j. In this case, i must be the parent region of j, $\mu(j) = i$ must hold.

Given a (proper) configuration W, we obtain a new (proper) configuration W' by executing the two phases of the transformations determined by the parallel application of maximal multisets of rules chosen for each compartment of the membrane system. A rule multiset is maximal, if it is applicable in a region, but if any other rule occurrence is added to it, then the sum of the left-hand sides is not contained in the region; that is, the extended rule multiset is not applicable any more. We call this a computational step and denote it by $W \Rightarrow W'$. A computation is a sequence of such computational steps, and it halts when a configuration is reached where there are no rules which can be applied. In such a halting configuration, the result of the computation appears as the contents of the output membranes.

We might consider additional features being present in the membrane system. First, we can add promoters and inhibitors to the rules. These are multisets of objects that regulate the rule applications in a way that the promoter $z \in \mathcal{M}(O)$ assigned to the rule r prescribes that z must be present in the region where the rule is applied, while the inhibitor $\neg z$ with $z \in \mathcal{M}(O)$ prevents the rule from being applied if z is present in the region.

Second, we can consider membrane dissolution. The set of objects is extended with an additional element δ that can appear on the right-hand sides of the rules. If δ appears in a rule r which is applied in the i-th region for some $i,\ 1 \leq i \leq n$, then the communication phase is executed as above, but after that, as the result of the presence of δ in region i, the region together with its set of rules R_i disappears from the P system. This means that the elements of region i are passed over to the parent region (except δ, which disappears), and the rules in R_i are not applied anymore. Note that the outermost region (the skin region) cannot dissolve.

Finally, we can consider a priority relation on rules, that is, a partial ordering (an antisymmetric and transitive relation) ρ_i on the set $R_i, 1 \leq i \leq n$. We say that r' has priority over r, or r' has higher priority than r (denoted as $r' > r$), if $(r', r) \in \rho_i$. In this case, if both r' and r were applicable in a configuration, then r is suppressed, that is, not allowed to be applied.

In Sect. 5, we show that all these features can be modeled by time Petri nets. The advantage of using time Petri nets is that the usual semantics, the usual unsynchronized way of firing of the transitions can be preserved: we do not inflict any additional condition on the transitions (like the requirement that the transitions fired in a computational step should constitute a maximal multiset).

2.2 Time Petri nets

In this section, following the definitions in [17] we define time Petri nets, a model rendering time intervals to transitions along the concept of [11]. First of all, we introduce the underlying place/transition Petri nets and then extend this model to the timed version.

A Petri net is a 5-tuple $U = (P, T, F, V, m_0)$ such that

1. P, T, F are finite, where $P \cap T = \emptyset$, $P \cup T \neq \emptyset$ and $F \subseteq (P \times T) \cup (T \times P)$,
2. $V : F \to \mathbb{N}_{>0}$,
3. $m_0 : P \to \mathbb{N}$.

The elements of P and T are called places and transitions, respectively, the elements of F are the arcs, and F is also called the flow relation of U. The function V is the multiplicity (weight) of the arcs, and m_0 is the initial marking. In general, a marking is a function $m : P \to \mathbb{N}$, it can be thought of as representing a state of the net U. We stipulate that for every transition $t \in T$, there is a place $p \in P$ such that $f = (p, t) \in F$.

Let $x \in P \cup T$. The pre- and post-sets of x, denoted by ${}^\bullet x$ and $x^\bullet$, respectively, are defined as ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$.

For each transition $t \in T$, we define two markings, $t^-, t^+ : P \to \mathbb{N}$ as follows:

$$t^-(p) = \begin{cases} V(p,t), & \text{if } (p,t) \in F, \\ 0 & \text{otherwise}, \end{cases}$$

$$t^+(p) = \begin{cases} V(t,p), & \text{if } (t,p) \in F, \\ 0 & \text{otherwise}. \end{cases}$$

A transition $t \in T$ is said to be enabled in the marking m, if $t^-(p) \leq m(p)$ for all $p \in {}^\bullet t$. Applying the notation $\triangle\, t(p) = t^+(p) - t^-(p)$ for $p \in P$, we define the firing of the transitions of a Petri net. A transition $t \in T$ can fire in m (notation: $m \longrightarrow^t$) if t is enabled in m. After the firing of t, the Petri net obtains the new marking $m' : P \to \mathbb{N}$ with $m'(p) = m(p) + \triangle\, t(p)$ for all $p \in P$, denoted as $m \longrightarrow^t m'$.

We obtain time Petri nets if we add time assigned to transitions of the Petri net. Intuitively, the time associated with a transition denote the last time when the transition was fired. We are considering only bounded time intervals. We present the definitions from [17], see also [18] for more information.

A time Petri net is a 6-tuple $N = (P, T, F, V, m_0, I)$ such that

1. the skeleton of N given by $S(N) = (P, T, F, V, m_0)$ is a Petri net, and
2. $I : T \to \mathbb{Q} \times \mathbb{Q}$ is a function assigning a rational interval to each transition, that is, for $t \in T$ and $I(t) = [\text{ eft }(t), \text{ lft }(t)]$, we have $0 \leq \text{ eft }(t) \leq \text{ lft }(t)$.

We call eft (t) and lft (t) the earliest and the latest firing times belonging to transition t, respectively.

Given a time Petri net $N = (P, T, F, V, m_0, I)$, a function $m : P \to \mathbb{N}$ is called a p-marking of N. Note that talking about a p-marking of N is the same as talking about a marking of $S(N)$. A state in N is a pair $u = (m, h)$, where h is a function called a transition marking (or t-marking) in N, $h : T \to \mathbb{R}_{\geq 0} \cup \{\#\}$. The two markings m and h satisfy the following properties.

For all $t \in T$,

1. if t is not enabled in m (that is, if $t^-(p) > m(p)$ for some $p \in {}^\bullet t$), then $h(t) = \#$,
2. if t is enabled in m (that is, if $t^-(p) \leq m(p)$ for all $p \in {}^\bullet t$), then $h(t) \in \mathbb{R}$ with $h(t) \leq \text{ lft }(t)$).

The initial state is the pair $u_0 = (m_0, h_0)$, where m_0 is the initial marking and for all $t \in T$,

$$h_0(t) = \begin{cases} 0, & \text{if } t \text{ is enabled in } m_0, \\ \#, & \text{otherwise.} \end{cases}$$

A transition $t \in T$ is ready to fire in state $u = (m, h)$ (denoted by $u \longrightarrow^t$) if t is enabled and eft $(t) \leq h(t) \leq$ lft (t).

Now we define the result of the firing for a transition that is ready to fire. Let $t \in T$ be a transition and $u = (m, h)$ be a state such that $u \longrightarrow^t$. Then the state u' resulting after the firing of t denoted by $u \longrightarrow^t u'$ is a new state $u' = (m', h')$, such that $m'(p) = m(p) + \triangle t(p)$ for all $p \in P$, and for all transitions $s \in T$, we have

$$h'(s) = \begin{cases} h(s), & \text{if } s \text{ is enabled both in } m \text{ and } m', \\ 0, & \text{if } s \text{ is not enabled in } m, \text{ but enabled in } m', \\ \#, & \text{if } s \text{ is not enabled in } m'. \end{cases}$$

Hence, the firing of a transition changes not only the p-marking of the Petri net, but also the time values corresponding to the transitions. If a transition $s \in T$ which was enabled before the firing of t remains enabled after the firing, then the value $h(s)$ remains the same, even if s is t itself. If an $s \in T$ is newly enabled with the firing of transition t, then

we set $h(s) = 0$. Finally, if s is not enabled after firing of transition t, then $h(s) = \#$.

Observe that we allow transitions to be fired several times in a row: if t is fired resulting in the new p-marking m', and t remains enabled in m', that is, $t^-(p) \leq m'(p)$ holds for all $p \in {}^\bullet t$, then the time associated with t does not change, $h'(t) = h(t)$.

Besides the firing of a transition, there is another possibility for a state to alter, and this is the time delay step. Let $u = (m, h)$ be a state of a time Petri net, and $\tau \in \mathbb{R}_{\geq 0}$. Then, the elapsing of time τ is possible for the state u (denoted $u \longrightarrow^\tau$) if for all $t \in T$ which are enabled in the current marking, the new time value assigned to t is less than the latest firing time of t. The state u', namely the result of the elapsing of time by τ denoted by $u \longrightarrow^\tau u'$, is defined as $u' = (m', h')$, where $m = m'$ and

$$h'(t) = \begin{cases} h(t) + \tau \leq \text{lft}(t), & \text{if } h(t) \neq \#, \\ \# & \text{otherwise.} \end{cases}$$

Note that the definitions ensure that we are not able to skip a transition when it is enabled: a transition cannot be disabled by a time delay step. This kind of semantics is called the strong semantics in [18]. Note also that classic Petri nets can be obtained as time Petri nets having $h(t) = [0, 0]$, as in such systems there are no time delay steps possible at all.

3 Connecting Petri nets and membrane systems

We start by discussing the time Petri net simulation of the basic variant of membrane systems we constructed in [1] and describe a simplification in the construction that we are going to use here. The model is based on the correspondence between Petri nets and membrane systems described in [9]. Membrane system configurations are represented by places (a, i) for each object a and membrane i, the number of tokens associated with such places correspond to the multiplicity of object a in membrane i. The rules of the membrane system are represented by transitions which move the tokens between the places according to the way that the rule application changes the corresponding object multiplicities in the membrane regions. A membrane system computation can be simulated by such a Petri net: if the membrane system halts, the Petri net reaches a configuration where no transitions can fire, and the marking of the Petri net corresponds to the halting configuration of the membrane system.

The new feature of the construction in [1] is the property that it does not change the usual Petri net semantics, and it simulates the maximal parallel rule application of membrane systems without requiring that the Petri net model fires its transitions in a maximal parallel manner. In general, both by membrane systems and by Petri nets, a computational step can be considered as a multiset of rules or as a multiset of transitions, respectively. In the case of Petri nets, an application of a multiset of transitions is maximal parallel, if augmenting the multiset by any other transition results in a multiset of transitions that cannot be fired simultaneously in the configuration represented by the current marking. In the case of membrane systems, maximal parallel execution means that no rule can be added to the multiset of rules such that the resulting multiset still forms a multiset of applicable rules. In our construction, the fireable transitions of the simulating Petri net can be executed in any order, and we do not impose any restrictions on the computational sequence of the Petri nets. This is possible, because we make an essential use of the time feature. (Note that the original place/transition Petri net model is not Turing complete, unlike the majority of the variants of symbol object membrane systems.)

In short, in [1] we have shown that given a membrane system Π (without priorities, membrane dissolution and promoters/inhibitors), there is a time Petri net N, such that N halts if and only if Π halts, and if they halt, then they provide the same result.

In the following, we will present analogous results also for membrane systems with extended features; that is, we show how time Petri nets can simulate membrane systems using rules with promoters/inhibitors, membrane systems with priorities assigned to the rules, and systems with the possibility of membrane dissolution.

In order to simplify the constructions, we modify the correspondence between membrane system computations and Petri nets, since we do not require that the simulating time Petri nets halt. For each halting configuration of the membrane system, the corresponding Petri net reaches a state where the computation continues in a cycle and the markings of places representing the membrane system configuration do not change any more.

In order to present this idea more formally, we say that a time Petri net $N = (P, T, F, V, m_0, I)$ simulates the computations of a membrane system Π by stabilizing in a configuration corresponding to a halting configuration of Π, if there is a subset of places $P_R \subseteq P$ and a subset transitions $T_R \subseteq T$, such that if and only if Π halts, N enters into a cyclic computation where

1. no transition of T_R is enabled,
2. the markings on the places of P_R do not change, and
3. the numbers of tokens associated with the places of P_R correspond to the halting configuration of Π.

Remark 1 In the following, we will show how to simulate halting computations of different membrane system variants with Petri net computations stabilizing in configurations

corresponding to halting membrane system configurations. However, the simulations also work if we apply our constructions to membrane systems which do not halt. In such cases, the Petri net simulation of the membrane system proceeds by a non-halting Petri net computation passing through an infinite sequence of configurations which corresponds to the infinite sequence of configurations of the membrane system. For more about the influence of different ways of halting on the computational power of P systems, also see the recent paper [4].

In the following, we first present a variant of the main theorem from [1] to introduce the basic ideas of the construction, and then we show how to modify this to be able to represent promoters/inhibitors, membrane dissolution, and the role of priorities on the rules.

Theorem 1 *For any membrane system Π (without priorities, membrane dissolution and promoters/inhibitors), there is a time Petri net N simulating the computations of Π by stabilizing in configurations corresponding to the halting configurations in Π.*

Proof Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$ be a membrane system and let $N = (P, T, F, V, m_0, I)$ be the corresponding Petri net. We define N so that a computational step of Π is simulated by two subnets of N. The two subnets correspond to the two computational phases of a computational step of a membrane system, namely the rule application and the communication phases. Let us see the construction in detail.

$$P = P_0 \cup \bar{P}_0 \cup \{p_{\text{app}}, p_{\text{com}}\}$$

with $P_0 = O \times \{1, \ldots, n\}$ and $\bar{P}_0 = \bar{O} \times \{1, \ldots, n\}$, where $\bar{O} = \{\bar{a} \mid a \in O\}$. For $1 \le i \le n$, the tokens $(a, i) \in P_0$ stand for the objects in the various compartments, while the tokens $(\bar{a}, i) \in \bar{P}_0 =$ represent the messages obtained in the course of the rule applications.

A marking $m : P \to \mathbb{N}$ corresponds to a configuration $(u_1, \ldots, u_n) \in \mathcal{M}(O)^n$ of Π as follows. If $m(p) = k$ for some $p = (a, i) \in P^0$, then there are k objects $a \in O$ in compartment i, that is, $u_i(a) = k$. If $m(\bar{p}) = l$ for some $\bar{p} = (\bar{b}, j) \in \bar{P}$, then there are l copies of object b which will enter into membrane j at the end of the computational step.

The presence of a token at the places p_{app} or p_{com} is enabling the rule application or the communication phases of the Petri net computation, respectively.

The initial marking corresponds to the initial configuration of $\Pi, m_0(p) = w_i(a)$ for every $p = (a, i) \in P_0,\ 1 \le i \le n$, and $m_0(\bar{p}) = 0$ for all $\bar{p} \in \bar{P}_0$. In addition, $m_0(p_{\text{app}}) = 1$ and $m_0(p_{\text{com}}) = 0$.

Let also

$$T = T_0 \cup \bar{T}_0 \cup \{t_{\text{com}}, t_{\text{app}}\}$$

where the transitions are defined as follows.

- For any rule $r \in R_i$ for some $i,\ 1 \le i \le n$, there is a transition $t_r \in T_0$ corresponding to r, and
- for any place $\bar{p} \in \bar{P}_0$, there is a transition $s_{\bar{p}} \in \bar{T}_0$.
- The transitions t_{com} and t_{app} transform the net from the rule application phase to the communication phase, and from the communication phase back to the rule application phase, respectively.

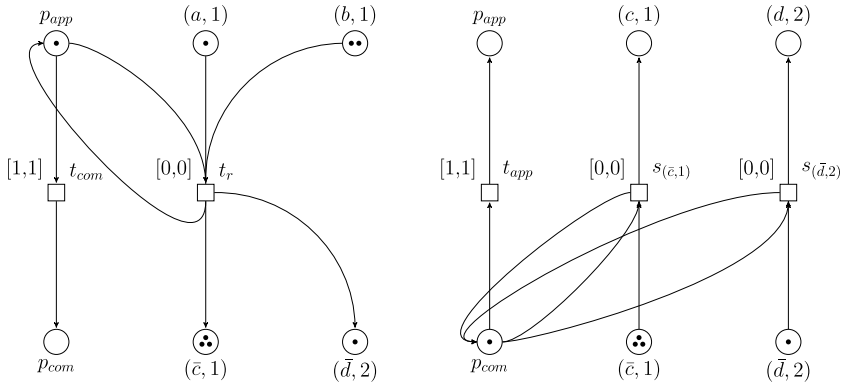
The input and output arcs and the time intervals associated with these transitions are as follows (see Fig. 1 for the graphical representation and an example).

- For each $t_r \in T_0$, $r \in R_i,\ 1 \le i \le n$, we have $(a, i) \in {}^\bullet t_r$ if and only if $a \in \text{lhs}(r)$, we have $(\bar{b}, j) \in t_r^\bullet$ if and only if either $(b, \text{in}_j) \in \text{rhs}(r)$ (where i is the parent region of j) or $(b, \text{out}) \in \text{rhs}(r)$ and region j is the parent region of i, or we have $(b, \text{here}) \in \text{rhs}(r)$ and $j = i$.
- Regarding the weights of the arcs, the weight of $f = (p, t_r) \in F$ for some $p = (a, i) \in P_0$ and $r \in R_0$ is the multiplicity of $a \in O$ on the left-hand side of r, namely $V(f) = \text{lhs}(r)(a)$. For $f = (t_r, \bar{p}) \in F$ where $\bar{p} = (\bar{b}, j) \in \bar{P}_0$, the weight of f is $V(f) = \text{rhs}(r)(b, \text{in}_j)$ if region j is a child region of i, $V(f) = \text{rhs}(r)(b, \text{out})$ if region j is the parent region of i, or $V(f) = \text{rhs}(r)(b, \text{here})$ if $i = j$.
- For each $s_{\bar{p}} \in \bar{T}_0$ with $\bar{p} = (\bar{a}, i) \in \bar{P}_0$ for some $1 \le i \le n$, we have $\bar{p} = (\bar{a}, i) \in {}^\bullet s_{\bar{p}}$ and $p = (a, i) \in s_{\bar{p}}^\bullet$.
- The weight of each of the arcs $f \in \{({}^\bullet s_{\bar{p}}, s_{\bar{p}}), (s_{\bar{p}}, s_{\bar{p}}^\bullet) \mid \bar{p} \in \bar{P}_0\}$ is $V(f) = 1$.
- The firing time intervals associated with all of the transitions $t \in T_0 \cup \bar{T}_0$ above are $I(t) = [0, 0]$.

In addition,

- we have $p_{\text{app}} \in {}^\bullet t \cap t^\bullet$ for all $t \in T_0$ and $p_{\text{com}} \in {}^\bullet t \cap t^\bullet$ for all $t \in \bar{T}_0$, all of these arcs having a weight of one, and
- we also have $p_{\text{app}} \in {}^\bullet t_{\text{com}}$, $p_{\text{com}} \in t_{\text{com}}^\bullet$, $p_{\text{com}} \in {}^\bullet t_{\text{app}}$, and $p_{\text{app}} \in t_{\text{app}}^\bullet$ with all of these arcs also having weight one.
- The firing time intervals associated with these transitions are $I(t_{\text{com}}) = I(t_{\text{app}}) = [1, 1]$.

To see how the above construction works, consider the following. A computational step of a membrane system is split into a rule application and a communication phase, and those two phases are simulated separately and in an alternating order. The simulation of the rule application phase is enabled by the presence of a token at p_{app}, and it



(a) Subnet for the rule application phase

(b) Subnet for the communication phase

Fig. 1 The subnet on the left simulates the application of $r : ab \to c^3(d, \text{in}_2)$ in region 1 (which is the parent region of 2) containing one a and two b objects. The figure shows the result of a single application of r. Transition t_r corresponding to the rule consumes an a and a b in region 1 and sends three tokens to the place $(\bar{c}, 1)$, and one token to $(\bar{d}, 2)$. This is in accordance with the fact that three objects c should be added to region 1, and one copy of d should be added to region 2 in the following communication phase. The subnet on the right simulates the communication phase of the membrane computational step. When the simulation of a maximal parallel rule application is finished, that is, when no transition associated with any rule is enabled any more, a time delay step can be performed, and then a token is passed over to p_{com} at time instance 1. Then, the transitions $s_{(\bar{c},1)}, s_{(\bar{d},2)} \in \bar{T}_0$ become enabled and ensure the correct placement of the tokens corresponding to the messages. When the updating of the marking is finished, a time delay step can follow, and then the token from p_{com} is passed back to p_{app} enabling the rule application phase once more

finishes only when no more rule applications are possible, ensuring that the rule application happens in the maximal parallel way. Any transition t_r corresponding to a rule r from some rule set of the membrane system can fire only if a token is found in p_{app}, but if no transition t_r can fire (that is, no rule r of the membrane system is applicable), then the transition t_{com} is enabled at time 1, so after performing a time delay step, a token is passed to p_{com}. The place p_{com} is connected to each transition $s_{\bar{p}} \in \bar{T}_0$ in both directions with an arc of multiplicity 1, which means that after the tokens have finished wandering back to their respective places (when no $s_{\bar{p}}$ is enabled any more), then there is still a token left in p_{com}. This token moves to p_{app} (after a time delay step) at time 1 via t_{app}, and the net is ready for simulating the next membrane computational step by performing a rule application phase of the simulation again.

When the Petri net arrives to a marking which corresponds to a halting configuration of the membrane system, no transitions in T_0 (and hence, no transitions in $\bar{T}_0$) are enabled, so it can only continue the cyclic computation consisting of passing a token back and forth between p_{app} and p_{com} without being able to change anything else in the distribution of the tokens, so in some sense, the representative of the halting membrane system configuration is "stabilized."

□

4 Extending the correspondence to membrane systems with more features

In this section, we examine the possibility of extending our core model to Petri nets that are able to represent various properties of membrane systems, such as the presence of promoters/inhibitors, membrane dissolution and a priority relation on the rules. The obtained Petri nets each build upon the basic model defined in the previous section, so, in most of the cases, we restrict ourselves to emphasize only the new elements of the constructions by which the basic Petri net model is extended.

We begin with discussing the case of promoters and inhibitors. We say that $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{P})$ is a membrane system with promoters/inhibitors, if $\mathcal{P}$ is mapping which maps the rules to $\mathcal{M}(O)^2$. Then $\mathcal{P}(r)$ is a promoter/inhibitor pair associated with $r \in R_i$ for some $i,\ 1 \le i \le n$, denoted as $(\text{prom}^r, \text{inhib}^r)$.

We say that a multiset $\mathcal{R}$ of rule occurrences is applicable in a configuration $(u_1, \ldots, u_n)$, if each of the following conditions is fulfilled. For all $1 \le i \le n$,

1. $\text{lhs}(r)(a) \cdot \mathcal{R}(r) \le u_i(a)$ for each $r \in R_i$ and $a \in O$,
2. $\text{prom}^r(a) \le u_i(a)$ for each $r \in \mathcal{R},\ r \in R_i,\ a \in O$, and
3. $\text{inhib}^r \not\subseteq u_i$, or in other words, there exists an $a \in O$, such that $u_i(a) < \text{inhib}^r(a)$ for all $r \in \mathcal{R},\ r \in R_i$.

Theorem 2 *For any membrane system Π with promoters/inhibitors, there is a time Petri net N simulating Π by stabilizing in configurations corresponding to the halting configurations of Π.*

Proof Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{P})$ be a membrane system with promoters/inhibitors. We construct N by extending the construction in the proof of Theorem 1, that is, the Petri net simulates the rule application and the communication phase separately in a similar manner. Here we concentrate on the differences concerning the rule application phase, the other elements of the construction can be adapted from Theorem 1.

Let $N = (P, T, F, V, m_0, I)$ be a time Petri net with places

$$P = P_0 \cup \bar{P}_0 \cup \{p_{\text{app}}, p_{\text{com}}\} \cup P_{\text{pro}} \cup P_{\text{inh}} \cup \{p_{\text{ini}}\}$$

where $P_0 \cup \bar{P}_0 \cup \{p_{\text{app}}, p_{\text{com}}\}$ are as in the proof of Theorem 1, and

$$P_{\text{pro}} = \{p_{\text{pro}(r)}, p_{\text{pro}(r)(a)}, p_{\neg\text{pro}(r)(a)} \mid r \text{ is a rule occurrence in } \Pi,\ a \in O\},$$

$$P_{\text{inh}} = \{p_{\text{inh}(r)}, p_{\neg\text{inh}(r)}, p_{\text{inh}(r)(a)}, p_{\neg\text{inh}(r)(a)} \mid r \text{ is a rule occurrence in } \Pi,\ a \in O\}.$$

The initial marking corresponds to the initial configuration of the membrane system as before. In addition, $m_0(p_{\text{ini}}) = m_0(p_{\neg\text{inh}(r)}) = m_0(p_{\neg\text{pro}(r)(a)}) \quad = m_0(p_{\neg\text{inh}(r)(a)}) = 1$ for all rules r, objects $a \in O$, while the rest of the places are marked with zero.

Let also

$$T = T_0 \cup \bar{T}_0 \cup \{t_{\text{com}}, t_{\text{app}}\} \cup T' \cup \{t_{\text{ini}}\}$$

where $T = T_0 \cup \bar{T}_0 \cup \{t_{\text{com}}, t_{\text{app}}\}$ are as int the proof of Theorem 1, and

$$T' = \{t_{\text{ini}(r)}, t_{\text{pro}(r)}, t_{\text{inh}(r)}, t_{\text{pro}(r)(a)}, t_{\text{inh}(r)(a)}, t_{\text{col}(r)}, t^1_{\text{col}(r)}, t^2_{\text{col}(r)}, t^3_{\text{col}(r)}, t_{\text{col}(r)(a)}, t^1_{\text{col}(r)(a)}, t^2_{\text{col}(r)(a)}, t^3_{\text{col}(r)(a)} \mid r \text{ is a rule occurrence in } \Pi,\ a \in O\}.$$

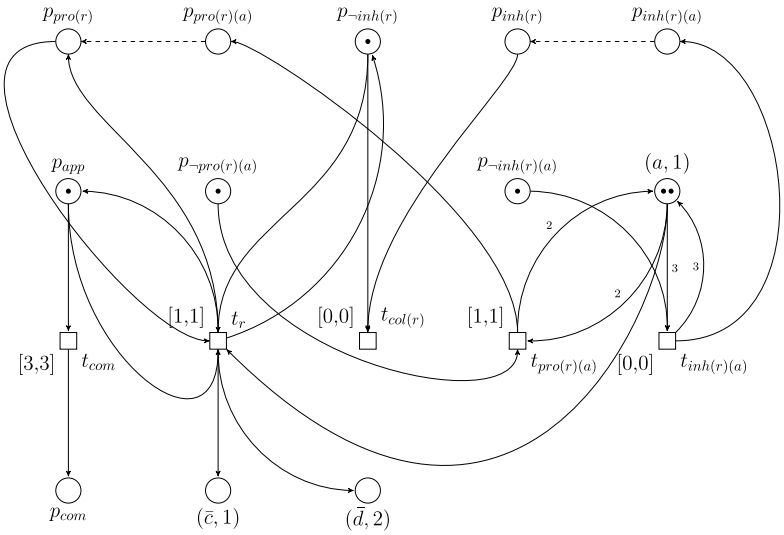


Fig. 2 The subnet simulating the rule application phase of a membrane system with promoters and inhibitors. The contents of the first region is a^2, and $r = a \to c(d, \text{in}_2)^3$ with $\text{prom}^r(a) = 2$, $\text{inhib}^r(a) = 3$ is applied. The rule application phase starts when $t_{\text{ini}(r)}$ sends a token to all places where one token is seen. First the inhibitor multiset is checked: if all $a \in \text{inhib}^r$ is present in the necessary number of occurrences (if the rule r should be inhibited), tokens are sent to each $p_{\text{inh}(r)(a)}$, and then to $p_{\text{inh}(r)}$. (The dashed arrows represent a collection of arcs and transitions described in the text: if the places $p_{\text{inh}(r)(a)}$ and $p_{\text{pro}(r)(a)}$ contain tokens for all $a \in \text{inhib}^r$ or $a \in \text{prom}^r$, then a token appears at $p_{\text{inh}(r)}$ or $p_{\text{pro}(r)}$, respectively.) Thus, if the rule should be inhibited, the token from $p_{\neg\text{inh}(r)}$ is removed, and t_r will not be able to fire. After the inhibitor, the promoter is checked: if all necessary objects are present, a token is sent to each $p_{\text{pro}(r)(a)}$, and then to $p_{\text{pro}(r)}$, which enables the firing of transition t_r corresponding to the execution of rule r. Since we have also made a time delay step before, the result of the promoter and inhibitor checks are kept the same until the repeated application of the allowed rules (simulating the maximal parallel application of the membrane system) is finished, and the communication phase begins. The subnet for the communication phase is not depicted here; it is similar to the subnet in Fig. 1, except for the transition t_{ini} which returns a token to p_{ini} (instead of p_{app}, as in the proof of Theorem 1)

The arcs connecting the places and transitions known from the construction of Theorem 1 are as before, with the exception of t_{app} which connects the places p_{ini} and p_{app}, and the new transition t_{ini} connecting p_{com} and p_{ini} (see Fig. 2 for the graphical representation and an example).

The input and output arcs associated with the new transitions, together with the new time intervals, are as follows.

Consider all $r \in R_i$ for some i, $1 \leq i \leq n$.

(a) For each $t_{\text{ini}(r)}$, we have $p_{\text{ini}} \in {}^\bullet t_{\text{ini}(r)}$ and $p_{\neg\text{inh}(r)}$, $p_{\neg\text{pro}(r)(a)}$, $p_{\neg\text{inh}(r)(a)} \in t^\bullet_{\text{ini}(r)}$.

(b) For each $t_{\text{pro}(r)}$, we have $p_{\text{pro}(r)(a)} \in {}^\bullet t_{\text{pro}(r)}$ for each $a \in O$, and $p_{\text{pro}(r)} \in t^\bullet_{\text{pro}(r)}$.

(c) For each $t_{\text{inh}(r)}$, we have $p_{\text{inh}(r)(a)} \in {}^\bullet t_{\text{inh}(r)}$ for each $a \in O$, and $p_{\text{inh}(r)} \in t^\bullet_{\text{inh}(r)}$.

The time intervals associated with the transitions described so far are [0, 0], while the weights of all the arcs above are one. The transitions defined in (*a*) initialize the rule application phase by putting a token to each of the necessary places after the appearance of a token in p_{ini} activates them.

The transitions described in (b)–(c) are represented by the dashed arrows in Fig. 2. They place a token to $p_{\text{pro}(r)}$ (or to $p_{\text{inh}(r)}$) if a token appears in $p_{\text{pro}(r)(a)}$ (or in $p_{\text{inh}(r)(a)}$) for each object a occurring in the promoter (or inhibitor) multisets associated with the rule r.

In addition to the above,

– for each $t_{\text{col}(r)}$, we have $p_{\neg\text{inh}(r)}, p_{\text{inh}(r)} \in {}^\bullet t_{\text{col}(r)}$ while $t^\bullet_{\text{col}(r)}$ is empty, and the time interval associated with $t_{\text{col}(r)}$ is [0, 0].

This transition disables the firing of t_r (disables the simulation of rule r) if r should be inhibited (which is signaled by the presence of a token in $p_{\text{inh}(r)}$).

– For each r, we have $p_{\neg\text{inh}(r)} \in {}^\bullet t^1_{\text{col}(r)}$, we have $p_{\text{inh}(r)} \in {}^\bullet t^2_{\text{col}(r)}$, we have $p_{\text{pro}(r)} \in {}^\bullet t^3_{\text{col}(r)}$. For each r and $a \in O$, we have $p_{\text{inh}(r)(a)} \in {}^\bullet t_{\text{col}(r)(a)}$, we have $p_{\neg\text{pro}(r)(a)} \in {}^\bullet t^1_{\text{col}(r)(a)}$, we have $p_{\neg\text{inh}(r)(a)} \in {}^\bullet t^2_{\text{col}(r)(a)}$, and we have $p_{\text{pro}(r)(a)} \in {}^\bullet t^3_{\text{col}(r)(a)}$. The time interval associated with all of these transitions is [2, 2].

The set of output places of the above described transitions are empty, they serve as "sinks," they remove unnecessary tokens at the end of the rule application phase.

For the rest of the arcs and their weights, together with the rest of the transitions and their associated time intervals, see Fig. 2.

The time Petri net described above alternates its rule application phase and communication phase to simulate the computational steps of the membrane systems. The alternation of the phases is guided by the cycling of a token from p_{ini} to p_{app}, from there to p_{com}, and then back to p_{ini}. Simultaneously with the arrival of the token to p_{app}, the rule application simulation subnets are also initialize for each rule, and then the checking of the presence of promoters and inhibitors start. The subnets for the rules which are not blocked after these checks can be executed, any of them repeatedly, until the available objects allow. (The results of the promoter and inhibitor checks are saved, so during the sequential simulation of a maximal parallel rule application step, rules can only become disabled when the objects on their left-hand sides are not available any more.) After the rule application phase is finished, the communication phase reorders the created objects to their goal regions as the system in the proof of Theorem 1 and in addition cleans up the garbage tokens left in the places which are used for the promoter/inhibitor checks of the rule application phase.

When the net arrives to a marking which corresponds to a halting configuration of the membrane system, then no rule application is possible, so no transitions can fire which are able to change the marking on the places corresponding to the membrane system configuration. This means that the representant of the halting membrane system configuration is "stabilized" in the sense of Sect. 3. □

Next, we turn our attention to membrane systems with dissolution. In such systems there exists a special object δ which can appear on the right sides of rules only. If such a rule is chosen in the actual maximal parallel rule application, then all the rules participating in that computational step are executed as usual, but after the maximal parallel step is over, the region where this rule was applied disappears, the objects wander into the parent region and the rules associated with it cease to operate.

Note that if a region i, $1 \leq i \leq n$ is dissolved during a computation, then not only the rules associated with i are disabled, but also some rules from other rule sets: those which contain the target indicator in_i on their right-hand sides. On the other hand, not all rules which send objects to region i are discarded. Those which are associated with a child region of i and contain the target indicator out continue to operate, but the objects they send "out" are sent to the parent region of i instead.

With this in mind, we construct a time Petri net simulating the operation of a membrane system with dissolution.

Theorem 3 *For any membrane system Π with dissolution, there is a time Petri net N simulating Π by stabilizing in configurations corresponding to the halting configurations of Π.*

Proof Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \delta)$ be a membrane system with dissolution. We construct a time Petri net

$N = (P, T, F, V, m_0, I)$ simulating Π. The construction again is an extension of the construction in the proof of Theorem 1. The rule application phase is very similar, except for the presence of new places corresponding to the appearance of the symbol δ symbolizing the dissolution of the membranes. The main difference manifests itself in the definition of the communication phase: we introduce a new phase, a "cleanup" phase for moving the elements of previously dissolved membranes to the parent regions. In addition, we also need to deal with blocking the application of certain rules which would send objects to the dissolved regions.

First we define the set of places.

$$P = P_0 \cup \bar{P}_0 \cup \{p_{\text{app}}, p_{\text{com}}\} \cup P' \cup \{p_{\text{cle}}, (\delta, i), (\bar{\delta}, i) \mid 1 \le i \le n\}$$

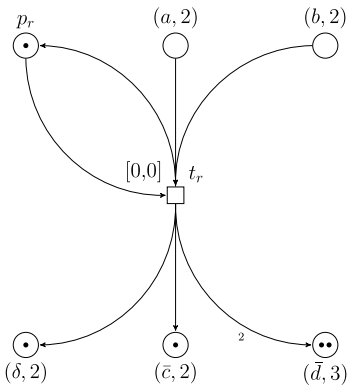
where $P_0 \cup \bar{P}_0 \cup \{p_{\text{app}}, p_{\text{com}}\}$ are as in the proof of Theorem 1, and

$$P' = \{p_r \mid r \text{ is a rule occurrence in } \Pi\}.$$

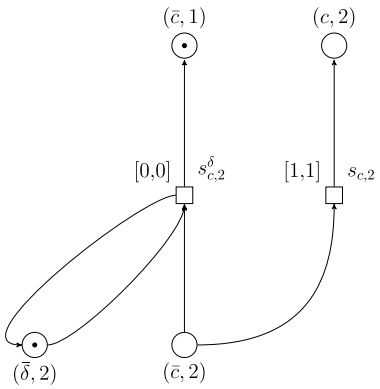
The initial marking corresponds to the initial configuration of Π with p_{app} also containing a token as before, and in addition, $m_0(p_r) \ge 1$ for all rules $r \in R_i$ for i, $1 \le i \le n$, the exact value depending on the number of ways r could be disabled due to the dissolving of membranes. More precisely, the value of $m_0(p_r)$ is the number of different target indicators of the form in_j on the right-hand side of rule r, or if this number is zero, then $m_0(p_r) = 1$. If p_r looses at least one of these tokens during the computation, then transition t_r corresponding to rule r will not be able to fire any more.

The role of the places $(\delta, i), (\bar{\delta}, i)$ is similar to the places of P_0 and $\bar{P}_0$: if the execution of some rule introduces δ in a region i, then a token arrives at (δ, i). Then, in the communication phase this token is moved to $(\bar{\delta}, i)$ where it remains, signaling to the rest of the system the fact that membrane i is dissolved.

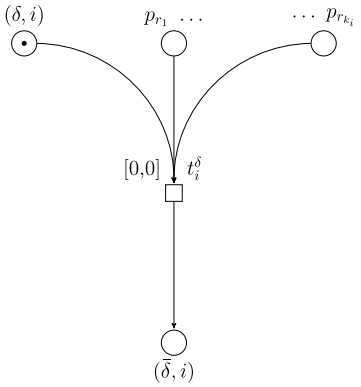
The other new place p_{cle} is used to signal the beginning of the cleanup phase, a new phase of the simulation when the contents of dissolved membranes are moved into the parent region.



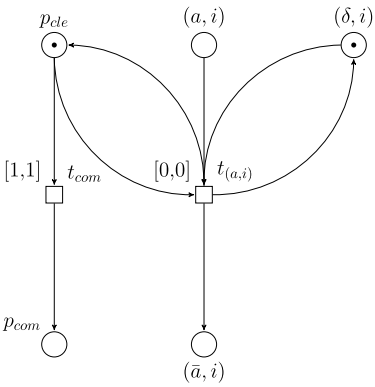
(a) Subnet for executing the rule $r : ab \to c(d, in_3)^2\delta$ in region 2: the membrane is dissolved



(b) Subnet for the communication phase: the token from $(\bar{c}, 2)$ moves to $(\bar{c}, 1)$ instead of $(c, 2)$



(c) Subnet for moving the token from (δ, i) to $(\bar{\delta}, i)$ while disabling rules by removing tokens from the corresponding places



(d) Subnet for emptying a dissolved membrane: $t_{(a,i)}$ prepares the objects in region i for transfer to the parent region

Fig. 3 The Petri net simulating a membrane system with dissolution. In **a**, **b**, the subnets corresponding to the application and communication phases of the rule $r : ab \to c(d, \text{in}_3)^2\delta$ are shown. When a membrane i is dissolved, the subnet in **c** is also activated before the communication phase. The subnet in **d** is active in the celanup phase which is between the rule application and the communication phases. It moves the tokens representing the contents of dissolved membranes to places from where they will be moved during the communication phase to places corresponding to the parent region

The transitions and the arcs are constructed as follows (see also Fig. 3 for a graphical representation of the most important subnets).

Let

$$T = T_0 \cup \bar{T}_0 \cup \{t_{\text{com}}, t_{\text{app}}\} \cup T^{\delta} \cup \{t_{\text{cle}}\}$$

where $T = T_0 \cup \bar{T}_0 \cup \{t_{\text{com}}, t_{\text{app}}\}$ has the same role as in the proof of Theorem 1, but some additional arcs are needed (see Fig. 3a).

– Each $t_r \in T_0$ where $r \in R_i$ for some i, $1 \le i \le n$, is also connected to the places p_r as $p_r \in {}^\bullet t_r \cap t_r^\bullet$ with arcs of weight $m_0(p_r)$. This has the effect that if the number of tokens is decreased in p_r, then the transition t_r simulating the execution of rule r cannot become enabled (making rule r disabled in the subsequent stages of the simulation). Moreover, if the rule r contains the dissolution symbol δ on its right-hand side, then t_r is also connected to the place (δ, i) as $(\delta, i) \in t_r^\bullet$ with an arc of weight one. The time interval associated with t_r is [0, 0].

The rest of the transitions are defined as follows. Let

$$T^{\delta} = \{t_p, s_p^{\delta} \mid \text{ for all } p \in P_0\} \cup \{t_i^{\delta} \mid 1 \le i \le n\}.$$

In addition to $s_p \in \bar{T}_0$, to execute the communication phase we also need the transitions s_p^{δ} for each $p \in P_0$ (see Figure 3(b)).

– For each i, $1 \le i \le n$ and $a \in O$, we have $(\bar{\delta}, i) \in {}^\bullet s_{(a,i)}^{\delta} \cap s_{(a,i)}^{\delta}{}^\bullet$ and $p_{\text{com}}, (\bar{a}, i) \in {}^\bullet s_{(a,i)}^{\delta}$, $(\bar{a}, j) \in s_{(a,i)}^{\delta}{}^\bullet$ where j is the parent membrane of i, with the weight of all these arcs being one, and the associated time interval being [0, 0]. If the communication phase is enabled by p_{com} and a token is present in $(\bar{\delta}, i)$, that is, if membrane i has been dissolved in some earlier step of the computation, then the tokens are moved from the places corresponding to objects in region i to the places corresponding to the same type of objects in the parent membrane. If there is no token in $(\bar{\delta}, i)$ (membrane i is not dissolved), then the transitions in $\bar{T}_0$ work as in the proof of Theorem 1, but to achieve the priority of $s_{(a,i)}^{\delta}$ over $s_{(a,i)}$, the later has an associated time interval of [1, 1].

The transitions t_i^{δ}, $1 \le i \le n$, are used to disable the rules with the target indicator in_i on their right-hand side when membrane i is dissolved (see Fig. 3c). The arcs are defined as follows.

– For each i, $1 \le i \le n$, we have $(\delta, i) \in {}^\bullet t_i^{\delta}$ and $(\bar{\delta}, i) \in t_i^{\delta\bullet}$. Moreover, for all rules r with at least one object (a, in_i) on their right-hand side for some $a \in O$, we have $p_r \in {}^\bullet t_i^{\delta}$. The time interval associated with these transitions is [0, 0]. When a token appears in (δ, i), that is, when membrane i is dissolved, the rules with target indicator in_i on their right-hand sides are disabled by removing tokens from the corresponding places. If a rule r could be disabled more than once because it contains more than one different types of such target indicators on its right-hand side, then the same number of tokens is assigned to p_r by the initial marking, so the functioning of the net is not blocked (it does not run out of tokens) when it tries to disable a rule which is already blocked.

The transitions t_p, $p \in P_0$ are used to perform a cleanup phase during which the objects of dissolved membranes are moved to places from where they will be transferred to the parent region during the following communication phase (see Figure 3(d)).

– For each i, $1 \le i \le n$ and $a \in O$, we have $p_{\text{cle}}, (a, i), (\delta, i) \in {}^\bullet t_{(a,i)}$ and $(\bar{a}, i) \in t_{(a,i)}^\bullet$ with all the arcs having weight one, and the transition having the associated time interval [0, 0]. The cleanup process is invoked once for each dissolved membrane when a token arrives to (δ, i), that is, when membrane i has been dissolved. After the initiation of the cleanup phase by a token at p_{cle}, the objects in places corresponding to membrane i are moved to the intermediate place from where they are transferred to the parent region of region i in the communication phase by the subnet already discussed above, depicted in Fig. 3b.

The functioning of the Petri net N is governed by a token circulating between the places p_{app}, p_{cle}, and p_{com}. Initially, there is a token in p_{app} which is moved to p_{cle} by transition t_{cle} in the time interval [1, 1] when the rule application phase is finished, that is, when there are no rule simulating transitions left which can fire. When the token is in t_{cle}, the objects of the regions that have been dissolved in the previous phase are prepared for being moved to the parent region. This is realized in the communication phase which is initiated when t_{com} moves the token further to p_{com}, the associated time interval being also [1, 1]. Finally, when all tokens representing the objects are in their correct places, the application phase is activated again by moving the token back to p_{app} by t_{app} at time interval [2, 2].

If no more rule application simulating transitions can fire, that is, when the corresponding configuration of the simulated membrane system is a halting configurations, then the only possible computation of the net is the cycling of the

token between p_{app}, p_{cle}, and p_{com}, thus, it has stabilized while representing a halting membrane system configuration. □

Finally, we tackle the problem of the representation of membrane systems with priorities in terms of Petri nets. Again, our construction is an extension of the core model presented in the proof of Theorem 1. We will modify the simulation of the rule application phase in order to account for the treatment of the new feature, but let us start with examining the role of priorities in membrane systems computations.

The system $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho)$ is a membrane system with priorities, if $\rho \subseteq \bigcup_{i=1}^{n}(R_i \times R_i)$. A rule $r \in R_i$, $1 \leq i \leq n$, is strongly applicable in configuration $(u_1, \ldots, u_n)$, if

1. r is applicable, that is, $lhs(r) \leq u_i$, and
2. for every $r' \in R_i$ such that $r' > r$, r' is not applicable.

Let $r_1, r_2 \in R_i$ be two rules of region i, and assume that $(r_1, r_2) \in \rho$, that is, $r_1 > r_2$. Then, considering a computational step, r_2 can be applied if r_2 is applicable and in addition, r_1 is not applicable (in the usual sense). For example, if $u_i = a^2b$, $r_1 = ab \to d$ and $r_2 = a \to c$, then the result of the maximal parallel step will be ad, instead of cd, since $r_1 > r_2$ and r_1 is applicable, which implies that r_2 cannot be applied in that maximal parallel step at all.

To simulate this behavior in our construction, we have to pick out the applicable rules and examine the priority relations in order to obtain the rules that are strongly applicable. We do this by stratifying the various tasks that have to be performed in the rule application phase with respect to time. Finding the strongly applicable rules takes place before the actual rule applications are simulated.

Theorem 4 *For any membrane system Π with priorities, there is a time Petri net N simulating the computations of Π by stabilizing in configurations corresponding to the halting configurations of Π.*

Proof Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho)$ a membrane system with priorities, and let us define the time Petri net $N = (P, T, F, V, m_0, I)$ as follows. We extend the net constructed in the proof of Theorem 1 by modifying the rule application phase. Before simulating the rule applications, we select those rules that are applicable, then deactivate the simulating subnets of those ones which are not strongly applicable. Then, the communication phase is simulated in the same way as in the proof of Theorem 1.

The set of places is defined as

$$P = P_0 \cup \bar{P}_0 \cup \{p_{\text{app}}, p_{\text{com}}\} \cup P' \cup P'_\rho \cup \{p_{\text{ini}}\},$$

where $P_0 \cup \bar{P}_0 \cup \{p_{\text{app}}, p_{\text{com}}\}$ are as in the proof of Theorem 1, and

$$P' = \{p_r^A, p_r^B \mid r \text{ is a rule occurrence in } \Pi\},$$
$$P'_\rho = \{p_{r_1>r_2}, \bar{p}_{r_1>r_2} \mid (r_1, r_2) \in \rho\}.$$

The new places keep track of the applicability and strong applicability of rules. A token in p_r^A for some rule r signals the applicability of r, while an additional token in p_r^B means that r is blocked by an applicable rule of higher priority, and the places $p_{r_1>r_2} \in P'_\rho$ contain a token if $r_1 > r_2$ holds according to ρ. (The role of the places of type $\bar{p}_{r_1>r_2}$ will be discussed later.)

The initial marking corresponds to the initial configuration of the membrane system, as before, but in addition, instead of p_{app}, the initial marking places a token in the new place p_{ini} which activates the first phase of the applicability check of the rules. Thus, $m_0(p_{\text{ini}}) = m_0(p_{r_1>r_2}) = 1$ for $r_1 > r_2$.

The transitions are defined as

$$T = T_0 \cup \bar{T}_0 \cup \{t_{\text{com}}, t_{\text{app}}\} \cup T' \cup \{t_{\text{ini}}\}$$

where $T_0 \cup \bar{T}_0 \cup \{t_{\text{com}}, t_{\text{app}}\}$ play the same role as in the previous constructions. The transitions in $\bar{T}_0$ are responsible for the simulation of the communication phase, they are defined exactly as before. The places t_{ini}, t_{app}, and t_{com} are responsible for initiating the different phases of the Petri net computation by moving a token along p_{ini}, p_{app}, p_{com}, and then back to p_{ini} again. The corresponding arcs are

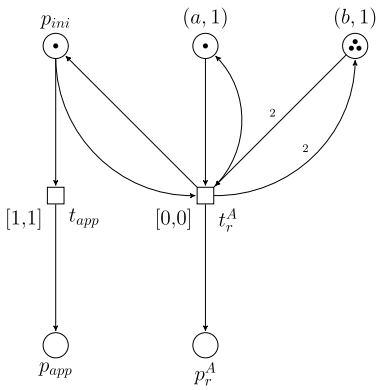


Fig. 4 Subnet for checking the applicability of a rule $r : u \to v \in R_1$ with $u = ab^2$. If at least one a and two bs are present in region 1, a token is sent to p_r^A

– $p_{\text{ini}} \in {}^\bullet t_{\text{app}} \cap t_{\text{ini}}{}^\bullet$, $p_{\text{app}} \in t_{\text{app}}{}^\bullet \cap {}^\bullet t_{\text{com}}$, and $p_{\text{com}} \in t_{\text{com}}{}^\bullet \cap {}^\bullet t_{\text{ini}}$, the associated intervals being $I(t_{\text{app}}) = I(t_{\text{ini}}) = [1, 1]$ and $I(t_{\text{com}}) = [4, 4]$.

The transitions in T' are

$$T' = \{t_r^A, t_{\text{col}(r)}^1, t_{\text{col}(r)}^2 \mid r \text{ is a rule occurrence in } \Pi\} \cup \{t_{r_1>r_2}, \bar{t}_{r_1>r_2} \mid r_1 > r_2 \text{ holds according to } \rho\}.$$

The first group of arcs connect t_r^A with the rest of the system. These transitions are activated by the presence of a token in p_{ini}, and once active, they check whether the corresponding rule is applicable or not. If all the objects on the left-hand side of r are available in the respective regions, a token is sent to p_r^A (see Fig. 4 for the graphical representation and an example).

– For each rule $r : u \to v \in R_i$ for some i, $1 \le i \le n$, if $u(a) = k > 0$ for some object $a \in O$, then $(a, i) \in {}^\bullet t_r^A \cap t_r^{A\bullet}$ and the weights of these arcs are k. In addition, $p_{\text{ini}} \in {}^\bullet t_r^A \cap t_r^{A\bullet}$, $p_r^A \in t_r^{A\bullet}$ and the time interval associated with t_r^A is [0, 0].

The subnet simulating the rule application phase is extended in order to be able to deal with priorities. Besides the arcs inherited from the construction in the proof of Theorem 1, the transitions of T_0 are augmented with a mechanism which disables the otherwise applicable rules which are suppressed by other applicable rules with higher priority. The transitions t_r are also connected to the places of P' and P'_ρ as follows. (An example with the graphical representation of the subnet can be seen in Fig. 5.)

– For any rule r, $p_r^A \in {}^\bullet t_r \cap t_r{}^\bullet$, thus, when the token is removed from p_r^A, transition t_r cannot fire.

When a token is present in $p_{r_1>r_2}$ (that is, when $r_1 > r_2$ holds) and also in both $p_{r_1}^A$ and in $p_{r_2}^A$, then $\bar{t}_{r_1>r_2}$ sends a token to $p_{r_2}^B$, enabling transition $t_{\text{col}(r_2)}$ which removes the token from $p_{r_2}^A$ disabling this way the firing of t_{r_2}. This is achieved by the following arcs.

– For any pair of rules $r_1 > r_2$, we have $p_{r_1>r_2} \in {}^\bullet \bar{t}_{r_1>r_2}$, $p_{r_1}^A, p_{r_2}^A \in {}^\bullet \bar{t}_{r_1>r_2} \cap \bar{t}_{r_1>r_2}{}^\bullet$, and $p_{r_2}^B, \bar{p}_{r_1>r_2} \in \bar{t}_{r_1>r_2}{}^\bullet$, with $\bar{t}_{r_1>r_2}$ having the associated time interval of [0, 0].
– For any rule r we have $p_r^A, p_r^B \in {}^\bullet t_{\text{col}(r)}^1$ with the associated time interval [1, 1], and $p_r^A \in {}^\bullet t_{\text{col}(r)}^2$ with time interval [3, 3].

The role of $t_{\text{col}(r)}^2$ is to remove the leftover tokens after the rule application phase.

– For any pair of rules $r_1 > r_2$, we also have $\bar{p}_{r_1>r_2} \in {}^\bullet t_{r_1>r_2}$, and $p_{r_1>r_2} \in t_{r_1>r_2}{}^\bullet$. With the associated time interval [3, 3], this transition returns the token to $p_{r_1>r_2}$ after the rule application phase is finished.

The places of $\bar{P}_0$ and the transitions of $\bar{T}_0$ for the communication phase are exactly the same as in the construction in the proof of Theorem 1, so we do not repeat them here.

□

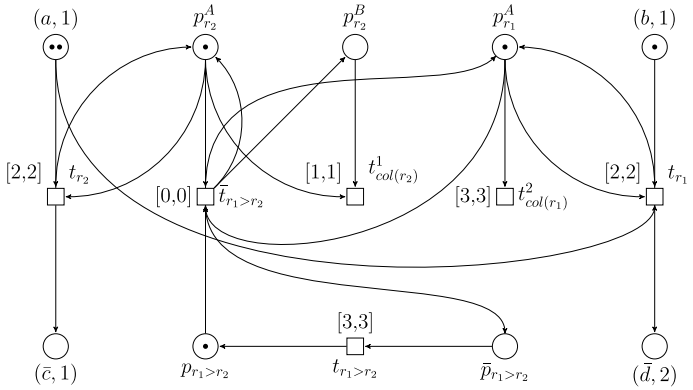


Fig. 5 The blocking of a rule due to the priority relation. Having a^2b in region 1 and $r_1, r_2 \in R_1$, $r_1 = ab \to d$, $r_2 = a \to c$ with $r_1 > r_2$, only t_{r_1} can fire: transition $\bar{t}_{r_1>r_2}$ delivers a token to $p_{r_2}^B$, then at time instance 1, the tokens from $p_{r_2}^A$ and $p_{r_2}^B$ are removed by transition $t_{\text{col}(r_2)}^1$. At time instance 3, the token from $\bar{p}_{r_1>r_2}$ is returned to $p_{r_1>r_2}$ by $t_{r_1>r_2}$, and the leftover tokens in places of type p_r^A (such tokens remain if the corresponding rules have not participated in the maximal parallel rule application) are cleaned up by transitions $t_{\text{col}(r)}^2$

5 Conclusions

In this paper, we have made a step forward in relating the membrane systems and Petri nets. We simulated membrane systems with promoters/inhibitors, membrane dissolution, and priority on the rules with time Petri nets, by further developing the Petri net model presented in [1]. We were able to simulate these sophisticated membrane computing features by preserving an important characteristic property of Petri nets; namely, the firings of the transitions can take place in any order: we do not impose any additional condition on the transition sequences in order to obtain the time Petri net model.

For a possible continuation of this line of research, it would be interesting to investigate the possibility of simulating membrane systems using other ways of synchronization, see [2, 4] for examples of such systems.

As another interesting direction for future research, we would also like to mention the descriptional complexity aspects of the presented simulations. It would be fruitful to conduct a comparative study on the sizes of the Petri net components necessary for simulating the different variants of membrane systems.

Acknowledgements György Vaszil was supported by Grant K 120558 of the National Research, Development and Innovation Office of Hungary (NKFIH), financed under the K 16 funding scheme. The work of Péter Battyányi was supported through the construction EFOP-3.6.3-VEKOP-16-2017-00002 by the European Union, cofinanced by the European Social Fund.

Funding Open access funding provided by University of Debrecen.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Aman, B., Battyányi, P., Ciobanu, G., & Vaszil, G. (2020). Local time membrane systems and time Petri nets. *Theoretical Computer Science*, *805*, 175–192.
2. Aman, B., & Ciobanu, G. (2019). Synchronization of rules in membrane computing. *Journal of Membrane Computing*, *1*, 233–240.
3. Desel, J., Reisig, W., & Rozenberg, G. (Eds.). (2004). *Lectures on concurrency and Petri nets. Lecture notes in computer science* (Vol. 3098). Berlin: Springer.
4. Freund, R. (2020). How derivation modes and halting conditions may influence the computational power of P systems. *Journal of Membrane Computing*, *2*, 14p–25.
5. Frisco, P. (2009). *Computing with cells: Advances in membrane computing*. Oxford: Oxford University Press.
6. Janicki, R., Lauer, P. E., Koutny, M., & Devillers, R. (1986). Concurrent and maximally concurrent evolution of nonsequential systems. *Theoretical Computer Science*, *43*, 213–238.
7. Kleijn, J., & Koutny, M. (2008). Processes of membrane systems with promoters and inhibitors. *Theoretical Computer Science*, *404*, 112–126.
8. Kleijn, J., & Koutny, M. (2009). A Petri net model for membrane systems with dynamic structure. *Natural Computing*, *8*, 781–796.
9. Kleijn, J. H. C. M., Koutny, M., & Rozenberg, G. (2006). Towards a Petri net semantics for membrane systems. In R. Freund, G. Păun, G. Rozenberg, & A. Salomaa (Eds.), *Membrane computing. WMC 2005. Lecture notes in computer science* (Vol. 3850, pp. 292–309). Berlin: Springer.
10. Kleijn, J., Koutny, M., & Rozenberg, G. (2006). Process semantics for membrane systems. *Journal of Automata, Languages, and Combinatorics*, *11*, 321–340.
11. Merlin, P.M. (1974). A study of the recoverability of computing systems. Ph.D. Thesis, University of California, Irvine.
12. Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, *61*(1), 108–143.
13. Păun, G. (2002). *Membrane computing, an introduction*. Berlin: Springer.
14. Păun, G., Rozenberg, G., & Salomaa, A. (Eds.). (2010). *The oxford handbook of membrane computing*. Oxford: Oxford University Press.
15. Peterson, J. L. (1981). *Petri net theory and the modeling of systems*. Upper Saddle River: Prentice Hall.
16. Petri, C. A. (1962). Kommunikation mit Automaten. Dissertation, Universität Hamburg.
17. Popova, L. (1991). On Time Petri Nets. *Journal of Information Processing and Cybernetics*, *27*(4), 227–244.
18. Popova-Zeugmann, L. (2013). *Time and Petri nets*. Berlin: Springer.
19. Reisig, W., & Rozenberg, G. (Eds.). (1998). *Lectures on Petri nets. Lecture notes in computer science, vols. 1491 and 1492*. Berlin: Springer.
20. Sosík, P. (2019). P systems attacking hard problems beyond NP: A survey. *Journal of Membrane Computing*, *1*, 198–208.
21. Zandron, C. (2020). Bounding the space in P systems with active membranes. *Journal of Membrane Computing*, *2*, 137–145.