Membrane Systems and Time Petri Nets

Bogdan Aman[1], Péter Battyányi[2], Gabriel Ciobanu[1], György Vaszil[2]

[1] Romanian Academy, Institute of Computer Science
Blvd. Carol I no. 8, 700505 Iaşi, Romania
`baman@iit.tuiasi.ro, gabriel@info.uaic.ro`

[2] Department of Computer Science, Faculty of Informatics
University of Debrecen
Kassai út 26, 4028 Debrecen, Hungary
`battyanyi.peter@inf.unideb.hu, vaszil.gyorgy@inf.unideb.hu`

Summary. We investigate the relationship of time Petri nets and different variants of membrane systems. First we show that the added feature of "time" in time Petri nets makes it possible to simulate the maximal parallel rule application of membrane systems without introducing maximal parallelism to the Petri net semantics, then we define local time P systems and explore how time Petri nets and the computations of local time P systems can be related.

1 Introduction

There has been several models applied for describing concurrency, communication and synchronization. Two of them are the graph-based model, known later as Petri nets, developed by C. A. Petri [11] and the tree-like model of embedded membranes called membrane or P systems invented by Gh. Păun [9].

Petri nets are state/transition systems: places are often used to contain information representing conditions in the system being modeled while transitions are used to represent events that can occur to modify the conditions. Some of the information, the input of the transition, is required for an event to happen, while some other information, the output of the transition, provides the result of the executed transition: they are the output of the transition. A Petri net is a bipartite graph: arcs point from input places to transitions and from transitions to places storing their outputs.

There may be some situations where the modelling of a system by conditions and events is not completely satisfactory, for example, when the assumption that all the transitions can take place in an arbitrary order does not describe the system correctly. To model the situation when time delay must be taken into account time Petri nets (TPN) were developed. Concerning time Peri nets, several models were elaborated: time was associated with transitions, places or arcs, etc. We consider the approach adopted by Merlin [8] rendering time to transitions. By this model, to every transition t we associate a closed interval $[a_t, b_t]$ such that a_t, $b_t \in \mathbb{Q}_{\geq 0}$. The transition can fire, if it is enabled and its local time $h(t)$ is such that $a_t \leq h(t) \leq b_t$. We adopt the strong semantics, which means that a transition which is enabled either must be fired at some point of the associated interval or it becomes non enabled by firing of another transition. In general, time Petri nets are more powerful than ordinary Petri nets, since time Petri nets are able to simulate Turing machines while, for ordinary Petri nets, this is not possible.

Membrane systems are parallel, distributed, synchronized models of computation where embedded membranes are organized in a tree like structure and computation takes place simultaneously in the different membranes in the forms of applications of rewriting rules. The rules evolve in a distributed manner: the application of a rule yields elements with labels, so called messages, which prescribe the exact place where the result of the rule application should move to. An element obtained by a rule application can either remain in the actual membrane, permeate to the parent membrane, or enter into one of its child membranes indicated by the rule. We consider here the basic model, that is, a membrane structure without dissolution rules. In addition we associate to each rule a time interval which gives a lower and an upper values for the time instance when the rule can be executed. We found technically simpler to consider every compartment as if a local stopwatch would operate in that compartment, though the same results could be obtained when we defined a global clock for synchronizing computational steps in the whole membrane system. We call our membrane systems local time membrane systems.

In this paper we relate local time membrane systems to time Petri nets such that the image Petri net of a membrane system by this mapping is suitable for answering questions in connection with the membrane system. For example, we can heavily lean on results in the area of time Petri nets concerning questions of reachability, which asks whether a certain configuration of the membrane system can be achieved, or threshold problems, where the question is whether a state can be reached from another in a certain time, or simply finding the paths requiring minimum/ maximum time between any two reachable states (see Popova-Zeugmann [14]).

There are several timed models for P systems in the literature (see [3], [4], [1]). The attempts for the simulation, up to the present, seem to take the approach similar to timed Petri nets ([6], [1], [2]), where certain values, the delay values, are assigned to rules. This means that the result of a rule application can appear only after that delay assuming a global clock synchronizes the computation of the system. Our model resembles much to that of time Petri nets: an interval is assigned to every rule and a clock local to each compartment synchronizes when the rule can be executed. A computational step is governed by a global clock: only when all membranes finish their action can a new step take place.

2 Membrane systems

First of all, we discuss some terminology used in the sequel. A finite multiset over an alphabet V is a mapping $M : V \to \mathbb{N}$ where $\mathbb{N}$ is the set of non-negative integers, and $M(a)$ for $a \in V$ is said to be the multiplicity of a in V. We say that $M_1 \subseteq M_2$ if for all $a \in V$, $M_1(a) \leq M_2(a)$. The union or sum of two multisets over V is defined as $(M_1 + M_2)(a) = M_1(a) + M_2(a)$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2)(a) = M_1(a) - M_2(a)$ for all $a \in V$. The multiset M can also be represented by any permutation of a string $w = a_1^{M(a_1)} a_2^{M(a_2)} \dots a_n^{M(a_n)} \in V^*$, where if $M(x) \neq 0$, then there exists j, $1 \leq j \leq n$, such that $x = a_j$. The set of all finite multisets over an alphabet V is denoted by $\mathcal{M}(V)$, the empty multiset is denoted by $\emptyset$ as in the case of the empty set.

A membrane system, or P system, is a tree-like structure of hierarchically arranged membranes embedded in the *skin* membrane as the outermost part of the system. Each region is delimited by a surrounding membrane, they can be arranged in a tree (cell-like [9]) structure or in a graph form (tissue-like [7] or neural-like [5]). In this paper we use the so-called symbol-object P systems [9] without dissolution, that is, each membrane has a label and enclosing a region containing a multiset of objects and rules and possibly some other membranes. The unique outer-most membrane is called the skin membrane. We assume the membranes are labelled by natural numbers $\{1, \dots, n\}$, and we use the notation m_i for the membrane with label i. Each membrane m_i, except for the skin membrane, has its parent membrane, which we denote by $\mu(m_i)$. As an abuse of notation we use μ both for the parent function and both for denoting the structure of the membrane system itself.

The contents of the regions of a P system evolve through rules associated with the regions. The computation of a P system is a locally asynchronous globally synchronous process: each multiset of objects in a region is formed locally by the rules attached to the regions, while a computational step of the whole system is a macro step: it finishes when all of the regions have finished their actions. In the variant we consider in this paper, the rules are multiset rewriting rules given in the form of $u \to v$ where u, v are multisets, and they are applied in a maximal parallel manner, that is, a region finishes its computation when no more rules can be applied in that computational step. In fact, the computational steps in the regions consist of two parts: first the rule application part and then comes a communication part where all the objects with labels find their correct places. The end of the computation of the system is defined by the following halting condition: a P system halts when no more rules can be applied in any of the regions; the result is a number, or a tuple of natural numbers- the number of certain objects in a membrane labelled as output.

Definition 1. *A* P system *of degree $n \geq 1$ is $\Pi = (O, \mu, w_1, \dots, w_n, R_1, \dots, R_n)$ where*

- *O is an alphabet of objects,*

- *μ is a membrane structure of n membranes,*
- *$w_i \in \mathcal{M}(O)$, $1 \le i \le n$, are the initial contents of the n regions,*
- *R_i, $1 \le i \le n$, 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 $v \in \mathcal{M}(O \times tar)$ where $tar = \{here, out\} \cup \{in_j \mid 1 \le j \le n\}$.*

Unless otherwise stated we consider the n-th membrane as the output membrane. A configuration is the sequence $W = (w_1, \dots, w_n)$ where w_k are the multiset contents of membrane m_k ($1 \le k \le n$). Let $\mathcal{R} = R_1 \cup R_2 \cup \dots \cup R_n$, where $R_i = \{r_{i1}, \dots, r_{ik_i}\}$ is the set of rules corresponding to membrane m_i. The application of $u \to v \in R_i$ in the region i means to remove the objects of u from w_i and to add the new objects specified by v to the system. The rule application in each region takes place in a non-deterministic and maximally parallel manner. This means that the rule application phase finishes, if no rule can be applied anymore in any region. As a result, each region where rule applications took place, is possibly supplied with elements of the set $O \times tar$. We call a configuration which is a multiset over $O \cup O \times tar$ an intermediate configuration. If we want to emphasize that $W = (w_1, \dots, w_n)$ consists of multisets over O, we say that W is a proper configuration. Rule applications can be preceded by priority check, if priority relations are present. Let $\rho_i \subseteq R_i \times R_i$ $1 \le i \le n$ be the (possibly empty) priority relations. Then $r \in R_i$ is applicable only if no $r' \in R_i$ can be applied with $(r', r) \in \rho_i$. We may also denote the relation $(r', r) \in \rho_i$ by $r' > r$. Priority relations will be mentioned only in Remark 2.

In the next phase the elements coming from the right hand sides of the rules of region i should be added to the regions as specified by the target indicators associated with them. If $rhs(r)$ contains a pair $(a, here) \in V \times tar$, then a remains in region i, this is the region where the rule is applied. If $rhs(r)$ contains $(a, out) \in V \times tar$, then a is added to the parent region of region i. In our membrane systems we assume that the results are formed in a designated membrane, the output membrane, of the system. Unless otherwise stated, we consider m_n as the output membrane of the system. If $rhs(r)$ contains $(a, in_j) \in V \times tar$ for some region j, then a is added to the contents of region j. In the latter case $\mu(m_j) = m_i$ holds.

3 The Petri net model

By defining a time dependent Petri net model we followed the definition proposed by Popova-Zeugmann [12] and chose a model rendering time intervals to transitions along the original concept of Merlin [8]. First of all, we define the notion of untimed Petri net and then extend this concept to the timed version.

Definition 2. *A Petri net is a 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 are called places and the elements of T are called transitions. The elements of F are the arcs and F is the flow relation of U. The function V is the multiplicity (weight) of the arcs and m_0 is the initial marking. We may occasionally omit the initial marking and simply refer to a Petri net as the tuple $U = (P, T, F, V)$. We stipulate that, for every transition t, there is a place p such that $V(p, t) \neq 0$.

In general, a marking is a function $m : P \to \mathbb{N}$. Let $x \in P$ or $x \in T$. The pre- and postsets of x, denoted by ${}^\bullet x$ and $x^\bullet$, respectively, are defined as ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$. Each arc has an incoming and outcoming multiplicity denoted as follows:

Definition 3. *Let t be a transition. We define below two markings, t^- and t^+, as multisets of places, which govern when a transition can be fired and how many tokens are added to the place p upon firing the transition, respectively.*

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

A transition is said to be enabled, if $t^-(p) \leq m(p)$ for all $p \in P$. Applying the notation $\triangle\, t = t^+ - t^-$, we are able to define a firing of the Petri net $U = (P, T, F, V)$.

Definition 4. *Let $U = (P, T, F, V, m_0)$ be a Petri net and let m be a marking in U. 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 will obtain the new marking m', where*

$$m' = m + \triangle\, t.$$

Notation: $m \longrightarrow^t m'$.

We obtain time Petri nets, if we add to the Petri net model information about time attached to transitions. Intuitively, the time associated to a transition will denote the last time when the transition or a transition with common preplace was fired. Though the definitions could be extended to unbounded time intervals also, we are concerned with bounded time intervals this time.

Definition 5. *A time Petri net (TPN) is a 6-tuple $N = (P, T, F, V, m_0, I)$ such that*

1. *the 5-tuple $S(N) = (P, T, F, V, m_0)$ is a Petri net,*
2. *$I : T \to \mathbb{Q}_{\geq 0} \times \mathbb{Q}_{\geq 0}$ and, for each $t \in T$, $I(t)_1 \leq I(t)_2$ holds, where $I(t) = [I(t)_1, I(t)_2]$.*

We call $I(t)_1$ and $I(t)_2$ earliest and latest firing times belonging to t, respectively. Notation: $eft(t)$, $lft(t)$.

A function $m : P \to \mathbb{N}$ is called a p-marking of N. Observe that talking about a p-marking of N is the same as talking about a marking of $S(N)$, where $S(N)$ is called the skeleton of N and, roughly speaking, it is the untimed Petri net obtained from N by omitting every reference to time.

Definition 6. *1. A transition marking (or t-marking) is a function $h : T \rightarrow \mathbb{R}_{\geq 0} \cup \{\#\}$.*

2. Let $N = (P, T, F, V, m_o, I)$ be a time Petri net, m a p-marking and h a t-marking in N. A state in N is a pair $u := (m, h)$ such that

a) $(\forall t \in T)(t^- \nleq m \rightarrow h(t) = \#)$,

b) $(\forall t \in T)(t^- \leq m \rightarrow h(t) \in \mathbb{R}_{\geq 0} \wedge h(t) \leq lft(t))$.

The initial state is the pair $u_0 = (m_0, h_0)$, where m_0 is the initial marking and

$$h_0(t) = \begin{cases} 0, & \text{if } t^- \leq m_0, \\ \# & \text{otherwise .} \end{cases}$$

Definition 7. *A transition t is ready to fire in state $u = (m, h)$ (in notation: $u \longrightarrow_t$), if t is enabled and $eft(t) \leq h(t)$.*

We define the result of the firing of a transition that is ready to fire.

Definition 8. *Let t be a transition and $u = (m, h)$ be a state such that $u \longrightarrow^t$. Then the result of the firing of t is a new state $u' = (m', h')$, such that $m' = m + \triangle t$ and*

$$h'(\hat{t}) = \begin{cases} h(\hat{t}), & \text{if } (\hat{t}^- \leq m, \hat{t}^- \leq m' \text{ and } {}^\bullet\hat{t} \cap {}^\bullet t = \emptyset) \textit{ or } t = \hat{t}, \\ \# & \text{if } \hat{t}^- \nleq m', \\ 0 & \text{otherwise .} \end{cases}$$

In words, the firing of a transition has multiple effects. First of all, it changes the t-marking of the system as it is customary by simple Petri nets. Moreover, the time values attached to the transitions may also change. If $\hat{t}$ was enabled before the firing of transition t and $\hat{t}$ remains enabled after the firing, moreover $\hat{t}$ has no common preplace with the transition which has just been fired, then the value $h(\hat{t})$ for $\hat{t}$ remains unchanged. The value $h(\hat{t})$ remains the same even if $\hat{t} = t$. If $\hat{t}$ is newly enabled with the firing of transition t or $\hat{t}$ has common preplace with t and $\hat{t}$ differs from t, then we have $h(\hat{t}) = 0$. If $\hat{t}$ is not enabled after firing of transition t, then $h(\hat{t}) = \#$.
Observe that we adopt a stronger condition for h to preserve the value for a transition $\hat{t}$ upon firing with transition t. We are not content with the fact that $\hat{t}$ should be newly enabled in order to have $h(\hat{t}) = 0$ in the subsequent computational step, but we also demand that t and $\hat{t}$ should not have common preplaces. To ensure multiple executions of the same transition, if $\hat{t} = t$, then $h(\hat{t})$ retains its value after the firing step.

Besides the firing of a transition there is another possibility for a state to alter, and this is the time delay step.

Definition 9. *Let t be a transition and $u = (m, h)$ be a state and $\tau \in \mathbb{R}^+$. Then elapsing of time with τ is possible for the state u (in notation: $u \longrightarrow^\tau$), if for all $t \in T$, $h(t) \neq \#$ implies $h(t) + \tau \leq lft(t)$. Then the result of the elapsing of time by τ is defined as follows: $u \longrightarrow^\tau u' = (m', h')$, where $m = m'$ and*

$$h'(\hat{t}) = \begin{cases} h(\hat{t}) + \tau, & \text{if } \hat{t}^- \leq m' \text{ for an arbitrary } \hat{t} \in T, \\ \# & \text{otherwise.} \end{cases}$$

Observe that the definition of the result of a time elapse ensures that we are not able to skip a transition when it is enabled: a transition cannot be made not enabled by a time jump. Finally, we define the notion of a feasible run in a time Petri net.

Definition 10. *Let $N = (P, T, F, V, m_o, I)$ be a time Petri net, assume $\sigma = t_1 \dots t_n$ is a sequence of transitions and $\tau = \tau_0\tau_1 \dots \tau_n$ ($\tau_i \in \mathbb{R}_{\geq 0}$) be a sequence of times. Then $\sigma(\tau)\tau_0 t_1 \tau_1 \dots t_n \tau_n$ is called a run. $\sigma(\tau)$ is a feasible run, if there are states $s = (m, h)$ and $s' = (m', h')$ such that $s \longrightarrow^*_{\sigma(\tau)} s'$. We may omit the argument τ from $\sigma(\tau)$ if it is clear from the context.*

Obviously, classic Petri nets can be obtained when $h(t) = [0, 0]$ for every transition and no time delay step is ever made.

4 Relating the Petri net model to the membrane system

First, we show how to establish a correspondence between the P system model without time and the model of time Petri nets. As the first step we give the underlying structure of the Petri net associated to a membrane system. The correspondence described below seems to have appeared first by Kleijn, Koutny and Rozenberg [6]. They define the correspondence by limiting the results of the Petri net computations only to those which can be obtained by a sequence of maximal parallel or maximally enabled transition steps. A step is a multiset of transitions and a transition is maximally enabled, if it is enabled and is not a proper subset of an enabled step. They establish a close correspondence between Petri nets with maximally enabled (max enabled) steps and membrane systems. Moreover, other semantics like locally enabled steps or minimal enabled steps could be considered.

In this case we preserve the original semantics for Petri nets: the fireable transitions can be executed in any order. This involves that we have to make essential use of the timed model, since ordinary Petri net model is not Turing complete in contrast to the general membrane system.

Definition 11. *Let $\Pi = (O, \mu, w_1, \dots, w_n, R_1, \dots, R_n)$ be a membrane system. Then we define the following places and transitions for the Petri net.*

1. *$P = P_0 \cup P_0^* \cup \{st_o, st_e, sem\}$, where $P_0 = V \times \{1, \dots, k\}$ and $P_0^* = V^* \times \{1, \dots, k\}$. We set $m_0(p) = w_j(a)$ for every place $p = (a, j)$. Intuitively, the places $V \times \{1, \dots, k\}$ correspond to the objects of V labelled by the indexes of the membranes and the places in $V^* \times \{1, \dots, k\}$ correspond to the objects on the right hand sides of R_i $(1 \leq i \leq n)$ labelled by messages. The places st_e, st_o, sem are additional places which serve for the synchronization of the Petri net model.*
2. *$T = T_0 \cup T_0^* \cup \{t_o, t_e, t^1_{sem}, t^2_{sem}\}$, where the sets of transitions T_0 and T_0^* are detailed in the subsequent parts of the definition and $\{t_o, t_e, t^1_{sem}, t^2_{sem}\}$ are auxiliary transitions to be specified later. Let $r_l \in R_i$, where $l \in \{1, \dots, n_{k_i}\}$.*

Then let t_l^i denote the transition corresponding to r_l and $T_0 = \{t_l^i \mid 1 \leq i \leq n, 1 \leq l \leq k_i\}$. A transition t_l^i connects elements of P_0 to P_0^*: if $p = (a, j)$, then $V(p, t_l^i) = lhs(r_l)(a)$, if $i = j$, and $V(p, t_l^i) = 0$ otherwise. Furthermore, if $p^* = (a^*, j)$, $V(t_l^i, p^*) = rhs(r_l)(a)$, if $i = j$, $V(t_l^i, p^*) = rhs(r_l)(a, out)$, if $j = parent(i)$ and $V(t_l^i, p^*) = rhs(r_l)(a, in_j)$, if $i = parent(j)$ and $V(t_l^i, p^*) = 0$ otherwise. Likewise, $T_0^* = \{s_j^i \mid 1 \leq i \leq k, 1 \leq j \leq n\}$ are such that $\{^\bullet(s_j^i) \mid 1 \leq i \leq k, 1 \leq j \leq n\} = P_0^*$, $\{(s_j^i)^\bullet \mid 1 \leq i \leq k, 1 \leq j \leq n\} = P_0$ and, if $a_i \in O$ and $1 \leq j \leq n$, then $p^* = (a_i, j)^*$ and $V(p^*, s_j^i) = V(s_j^i, (a_i, j)) = 1$ and all the other values are 0. The transitions t_{sem}^1 and t_{sem}^2, belonging to the semaphore, will be treated in the section.

3. The intervals belonging to the elements of $T = T_0 \cup T_0^*$ are $[0, 0]$, the transitions aiming for the synchronization have various time intervals to be specified later.

In words, we simulate the rule $r_l \in R_i$ with transition t_l^i such that the weights of the arcs reflect the multiplicities of the elements in compartment i, and the transitions s_j^k ensure the correct reordering of the elements with messages when the rewriting phase is finished. If we term the rule application phase as the odd and the communication phase as the even part of the operation, we obtain two Petri nets for the subsequent phases of the simulation, which are illustrated in Figures 1 and 2. A little more detailed, the complete Petri net acts as follows. The two main sets for the places correspond to the objects of the membrane system. If $a_i \in V$ has k_i occurrences in m_j, then, for $p = (a_i, j)$, $m(p) = k_i$. Likewise, assume at the end of a rule application phase we have k_i' occurrences of $(a_i, here)$ in m_j, and k_i'' occurrences of (a_i, out) in m_l, where $j = parent(l)$ and k_i''' copies of (a_i, in_j) for $l = parent(j)$, then $m(p^*) = k_i' + k_i'' + k_i'''$, where $p^* = (a_i, j)^*$. At the rule application phase the element st_o controls the process: if there are any transitions that are enabled, then they are executed. Otherwise a time elapse is applied and a token from st_o is passed over to sem at time 1. The situation is similar with the communication phase: if every element has found its correct place, then no more transition s_i^j is possible and st_e gives control to sem by passing a token to sem at time instance 1.

We ensure the globally asynchronous locally synchronous character of the membrane system for the Petri net by defining a semaphore which governs the distinct groups of membrane transitions, like rewriting phase, where objects are replaced in accordance with rewriting rules, or communication phase, where objects labelled with tags in_j, *here, out* find their correct places.

In what follows we define the timed part of the Petri net that provides the synchronization.

Assume the semaphore is denoted by the tuple $Sem = (\{sem\}, R, F_{sem}, V_{sem}, I)$. In some sense the semaphore divides the rule application and communication parts of the operation of the P system. The place *sem* of the semaphore is the place where this choice takes place. The place *sem* obtains either one or two tokens.

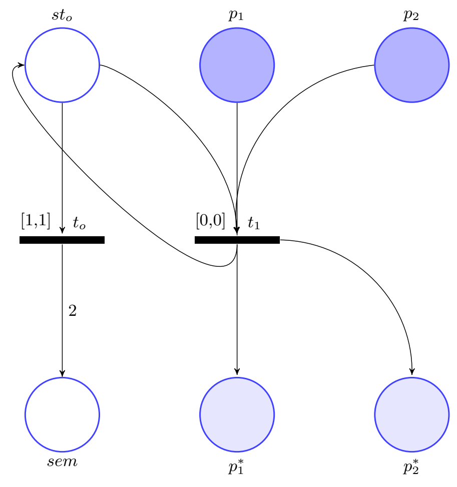

Fig. 1. The Petri net simulating the rule application part of a membrane computational step.

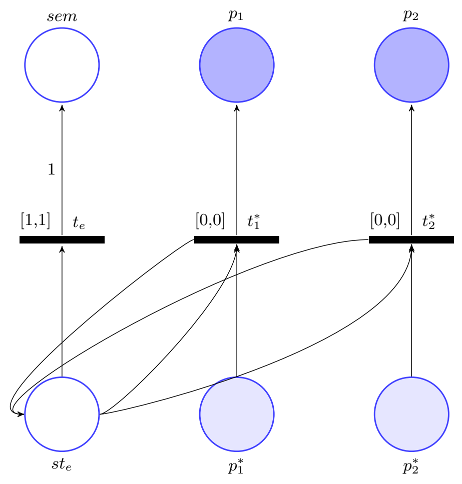

Fig. 2. The Petri net simulating the communication part of a membrane computational step.

If the odd phase is finished, then the semaphore obtains 2 tokens and otherwise, from the even phase, it obtains 1. One of the transitions require 2 tokens with time interval $[0,0]$- this transitions leads to st_e, and the other one requires 1 token with time interval $[0,0]$. The latter transition points to st_o. This means that two tokens enable the semaphore to activate the even phase, on the other hand, if at the end of the even phase it receives back only one token, then, after a time jump of 1, only the odd phase can be activated. We illustrate the semaphore in Figure 3.

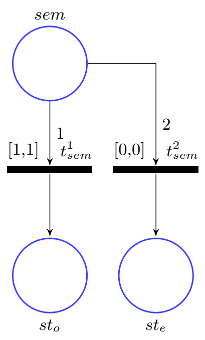

Fig. 3. The semaphore for the Petri nets.

By this, we have simulated a membrane system with a time Petri net such that in the Petri net model no restriction on the transitions is made: every transition which is ready to fire can be fired in any order. We summarize the above considerations in the following theorem.

Theorem 1. *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$ be a membrane system. Then there exists a time Petri net $N = (P, T, F, V, m_0, I)$ defined as in Definition 11 such that, for any computational sequence W of Π yielding an output, there exists a feasible run of N yielding the same output as W.*

We remark that, by keeping the core construction, it is not difficult to adjust the Petri net model so that it is able to simulate membrane systems defined with semantics other than the maximal parallel semantics. As a short remark, we consider lmax-parallelism (or locally max-enabledness) that was treated by Koutny and Kleijn and Rozenberg [6]. A computational step is lmax-parallel in a membrane system, if, for every membrane, the rules of the compartment are executed in a maximally parallel manner, or no rule of that membrane is executed at all in that computational step. To handle lmax-parallelism, Koutny and Kleijn and Rozenberg introduced localities for a Petri net. We define the notion of Petri nets with localities in accordance with the definition of Koutny and Kleijn and Rozenberg [6].

Definition 12. *A Petri net with localities (in short PNL) is a tuple* $NL = (P, T, V, D, m_0)$*, where* P*,* T*,* V *and* m_0 *are as defined by the base model and* $D : T \to \mathbb{N}$ *is a locality mapping. As in the original model, we assume that, for every transition* t *there is a place* p *such that* $V(p,t) \neq 0$*.*

The reason for introducing localities in [6] was the intention of simulating computation distributed to compartments in a membrane system. In the language of Petri nets with localities this is achieved when we define a computational step U as a multiset of transitions so that $l \in D(U)$ implies that U is maximal with respect to the transitions with $D(t) = l$. In our timed model we achieve lmax parallelism by inserting a nondeterministic choice at the beginning of every odd round: either a maximal parallel computational step is simulated concerning region l, or no computation takes place with respect to that region in the underlying computational step. We omit the details.

As a final remark, we stress out that our simulating Petri net works in a one-step manner in contrast to the model defined by Kleijn, Koutny and Rozenberg [6]. This means that exploring the state space seems to be an easier task by this model- there is no need to maintain a huge stack for keeping track of the possible successors of the present state obtained by a maximal parallel step. Hence, from the practical point of view using a time Petri net model for a membrane system seems to be promising.

5 Local time membrane systems

In this section we associate time to the rules of P systems and present a translation from local time P systems into time Petri nets such that the durations of the membrane computational steps can be estimated with the elapsed time in the computation of the time Petri net. This allows us to formulate several properties of local time membrane systems based on the well developed theory of time Petri nets. In the first part of the section we formulate the necessary definitions what we mean by local time membrane systems and elapsed time in a local time membrane system, then we present the simulation of membrane systems by Petri nets.

Definition 13. *We define the notion of a local time P system as a P system together with functions* $\mathcal{I} : R \to IntQ$ *and* $\mathcal{T} : \{1, \ldots, n\} \to \mathbb{R}_{\geq 0}$ $(1 \leq i \leq n)$*, where* R *is the set of all rules and* $IntQ$ *is the set of all closed intervals with nonnegative rational endpoints and there are* n *compartments of* Π*. The value* $\mathcal{T}(i)$ *is called the local time for the* i*-th membrane. A configuration of a local time P system is a pair* $(W, \mathcal{T})$*, where* W *is the configuration as a multiset of objects and messages as before and* $\mathcal{T}$ *is stands for the local time function. We may write* $\mathcal{T}_i$ *for* $\mathcal{T}(i)$ *in the sequel.*

Observe that, since $IntQ$ denotes the intervals with rational endpoints, we may assume that the endpoints of the intervals belonging to the rules are integers.

We have to multiply with the least common multiplier of the nominators of the endpoints in any other case. Likewise for the case of the time Petri nets. Next we define a computational step in a local time P system.

Definition 14. *Let $\Pi = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time P system. A run is a tuple $\sigma = (\sigma_1, \ldots, \sigma_n)$, where $\sigma_i = \tau_1^i r_1^i \ldots \tau_{k_i}^i r_{k_i}^i \tau_{k_i+1}^i$ such that $\tau_j^i \in \mathbb{R}_{\geq 0}$ and $r_j^i \in R_i$ for $1 \leq j \leq k_i$ and $1 \leq i \leq n$. We call the element σ_i the i-th selection. Moreover, we stipulate that the rules in σ_i form a maximal parallel multiset of rules. The elapsed time for σ_i is the sum of the τ-s in the selection.*

Remark 1. It seems to us that instead of defining local times for each compartment separately we could have chosen to give a global clock for the whole membrane system when talking about a computational step. The definition with a global clock could be given in such a way that not only the evolving of the system but every quantitative property, like elapsed time during a computational step, minimal/maximal time between two configurations, would remain the same. Technically, it seems to be a clearer formulation to introduce a local clock in each membrane, though.

Next we describe how the system can evolve during a selection belonging to a membrane. To facilitate the treatment we shall talk about the configuration of membrane i, as well, not only about a configuration of the whole system.

Definition 15. *Let m_i be a compartment with (possible) intermediate configuration w_i. Then $(w_i, \mathcal{T}_i)$ is the (timed) configuration of m_i.*

In what follows, when we talk about a configuration of a local time membrane system, we mean a timed configuration of the system, unless otherwise stated. If we want to emphasize that we talk about configurations without the time component, then we will use the term object configuration. The computation in a compartment can evolve through two steps when we consider a selection.

Definition 16.
1. *rule execution: Assume $r \in R_i$, for some $1 \leq i \leq n$, is enabled, that is, $lhs(r) \leq w_i$, where $(w_i, \mathcal{T}_i)$ is the configuration for m_i. Moreover, assume r is ready to be executed, which means r is enabled and $\mathcal{T}_i \in [\mathcal{I}(r)^-, \mathcal{I}(r)^+]$. Then $(w_i, \mathcal{T}_i) \longrightarrow_r (w_i', \mathcal{T}_i)$, where $w_i' = w_i - lhs(r) + rsh(r)$.*
2. *time elapse: Let $\tau \in \mathbb{R}_{\geq 0}$. Then we distinguish two types of semantics:*
   a) weak semantics: $(w_i, \mathcal{T}_i) \longrightarrow_r (w_i, \mathcal{T}_i')$ and $\mathcal{T}_i' = \mathcal{T}_i + \tau$,
   b) strong semantics: $(w_i, \mathcal{T}_i) \longrightarrow_r (w_i, \mathcal{T}_i')$ and $\mathcal{T}_i' = \mathcal{T}_i + \tau$ only if, for every $r \in R_i$, $lhs(r) \leq w_i$ and $\mathcal{T}_i \leq \mathcal{I}(r)^+$ implies $\mathcal{T}_i + \tau \leq \mathcal{I}(r)^+$.

A configuration $(w, \mathcal{T})$ can evolve in two ways. If a rule $r \in R_i$ is enabled and, furthermore, the local time for m_i is such that $\mathcal{T}_i$ lies in the interval $[\mathcal{I}(r)^-, \mathcal{I}(r)^+]$, then r can be executed. Rule execution in a membrane does not change the time part of a configuration, it changes only the multiset of objects in the underlying membrane. The second possibility is time elapse. If we adopt the weak semantics, then the time elapse step can take place at any time instance. This may involve that a rule which was ready to be executed before the time elapse step can be no more executed after that step. When we adopt the strong semantics this situation is impossible. In other words, the strong semantics does not allow us to skip rules that are enabled and their time interval makes their application possible by a time jump. The weak semantics permits us to apply an arbitrary choice of the rules. As usual, computations in the membranes evolve by a maximal parallel manner. When every membrane has finished its operation, the communication phase of the computational step begins, as by the non-timed case. The new computational step starts with a configuration $(W, \mathcal{T}_0)$, where $\mathcal{T}_0(i) = 0$ $(1 \leq i \leq n)$. By a computation, we understand a sequence of subsequent proper configurations, that is, configurations with no objects labelled by messages. When the computation halts, the result is stored in the output membrane, as before.

6 Local time P systems and time Petri nets

Next we give simulations for local time membrane systems where the elapsed time is the sum of the time jumps in a selection and the elapsed time for a run or a computational step is the maximum of these sums when we consider the different compartments. We perform the simulation for both the weak and strong semantics of local time membrane systems.

First of all, we deal with the case of a membrane system with the weak semantics. The underlying model is the same as in Definition 11, and we try to simulate membrane systems of the weak semantics by time Peri nets equipped with the strong semantics. We identify rule applications with transitions as before. In order to keep the local nature of the Petri net, we split our model into subnets corresponding to the execution in the various compartments. The main idea is that we divide the possible time duration of the computation in a membrane membrane into time intervals of length 1, and, concerning these time intervals, a nondeterministic choice of the transitions which are ready to fire is considered. The transitions in the same group are assumed to take place in the time interval $[0, 0]$, $[0, 1]$ or $[1, 1]$, a group of transitions is initiated by a place q_j^i as illustrated in Figure 4. When operating with a group of transitions, we may apply time elapse between firing steps up to the point when we fall into the next group of transitions. The passage is provided by transitions n_j^i in Figure 4. Let us apply the following notation in the definition below. Assume B is the least integer greater than the right hand sides of the intervals $\mathcal{I}(r)$. Let m_i be a membrane. Let

$$R_j^i = \{r \in R_i \mid j \in [eft(r), lft(r)]\}$$

for $0 \leq j \leq B - 1$, $j \in \mathbb{N}$ and $1 \leq i \leq n$. Then $\cup_{i=1}^{n} \cup_{j=0}^{B-1} R_j^i = R$. Observe that the distinct sets R_j^i may overlap for different j-s.

Definition 17. *Let $\Pi = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system with the weak semantics. Assume B is the least integer greater than the right hand sides of the intervals $\mathcal{I}(r)$. We define the associated time Petri net as follows.*

1. *The main constituents of P are the sets P_0, P_0^*, Q_i $(1 \leq i \leq n)$, where $P_0 = V \times \{1, \ldots, n\}$, $P_0^* = V^* \times \{1, \ldots, n\}$ are as in Definition 11. The computational step is governed by the set of places $Q_i = \{q_j^i \mid 0 \leq j \leq B, 1 \leq i \leq n\}$. There are some additional places including the places for the semaphore, places to mark the end of the computation in the simulated compartment, and places which lead to nonterminating computations. The semaphore is the same as in the previous models. Moreover, since the computational process in every compartment evolves separately, this is modelled by the Petri net: there are sub Petri nets the computations in which are triggered in a distributive manner. The operation of the sub Petri net corresponding to membrane m_i is initiated by transferring a token to the places q_0^i. At the end of the operation of the sub Petri net the place fin_i obtains a token and waits for the rest of the sub nets simulating the other compartments to finish computing. As usual, P_0 represents the actual configuration of the membrane system. We set $m_0(p) = w_j(a)$ for every place $p = (a, j)$. As before, the places in $V \times \{1, \ldots, n\}$ correspond to the objects on the left hand sides of R_i $(1 \leq i \leq n)$ of the membrane rules, while the elements of $V^* \times \{1, \ldots, n\}$ correspond to the objects on the right hand sides of the membrane rules labelled by messages.*
2. *As before, T_0 correspond to the membrane rules, while the transitions of T_0^* ensure the simulation of the communication phase of a membrane computational step. Let $r_l \in R_i$, where $l \in \{1, \ldots, n_{k_i}\}$. Then T_0 formed by t_l^i $(1 \leq i \leq n, 1 \leq l \leq k_i)$ and $T_0^* = \{s_j^i \mid 1 \leq i \leq k, 1 \leq j \leq n\}$ are defined as in Definition 11 with the slight modification as follows. Let $r \in R_j^i$, assume t is the transition associated with r in T_0. Then $(q_j^i, t) \in F$, $V(q_j^i, t) = 1$ and $(t, q_j^i) \in F$, $V(t, q_j^i) = 1$ $(1 \leq j \leq B-1, 1 \leq i \leq n)$. Moreover, for each each q_j^i $(1 \leq j \leq B-1)$ we have transitions n_j^i and t_{Bj}^i such that $(q_j^i, n_j^i) \in F$, $(n_j^i, q_{j+1}^i) \in F$ and $(q_j^i, t_{Bj}^i) \in F$, $(t_{Bj}^i, q_B^i) \in F$ with multiplicities 1. Furthermore, every transition in $t \in T_0$ corresponding to some $r \in R_i$ has a second copy $\widetilde{T}_0$, which is connected to q_B^i by $(q_B^i, \tilde{t}) \in F$ and, for every $p \in P_0$, we have an arrow pointing to $\tilde{t}$ if and only if $(p, t) \in F$ with the same multiplicity as that of (p, t). There is a state $perp_{\tilde{t}}$ for every $\tilde{t}$ which induces an infinitely working sub Petri-net. Finally, there is an arrow from q_B^i pointing to fin_i through transition t_{fin_i}. The places fin_i lead to sem through a transition fin_o. When we consider the case of T_0^*, it is enough to apply one auxiliary place, say fin_e, to check whether every transition in T_0^* has finished its operation: $(fin_e, \tilde{t}) \in F$, $(\tilde{t}, fin_e) \in F$ and $(fin_e, t_{fin_e}) \in F$, $(t_{fin_e}, sem) \in F$.*
3. *As to the timings, let $r \in R_j^i$, assume t corresponds to r in the Petri net. If $j = lft(r)$ then let $\mathcal{I}(t) = [0, 0]$, and if $[j, j+1] \subseteq [eft(r), lft(r)]$, then $\mathcal{I}(t) = [0, 1]$, else, if $j = eft(r)$, then $\mathcal{I}(t) = [1, 1]$. Let the members of T_0^**

have intervals $[0,0]$. *For the other transitions:* $\mathcal{I}(n_j^i) = [0,1]$, $\mathcal{I}(\tilde{t}_j) = [0,0]$ *and* $\mathcal{I}(t_{fin_i}) = [1,1]$. *The semaphore is the same as in the core model.*

Figure 4 presents a snapshot of the model: it illustrates the Petri net assigned to membrane m_i.

Fact 2 *Let* $r \in R^i_{j_{min}}, \ldots, r \in R^i_{j_{max}}$, *let* $j_{min} \leq j \leq j_{max}$, *assume* $[i_t^-, i_t^+]$ *is the interval belonging to* t *in the sense of Definition 17 when considering* $r \in R^i_j$, *where* t *is assigned to* r. *Then* $\mathcal{I}(r)^- \leq i_t^- + j \leq i_t^+ + j \leq \mathcal{I}(r)^+$. *Moreover,* $i_t^- + j_{min} \leq \mathcal{I}(r)^-$ *and* $\mathcal{I}(r)^+ \leq i_t^+ + j_{max}$.

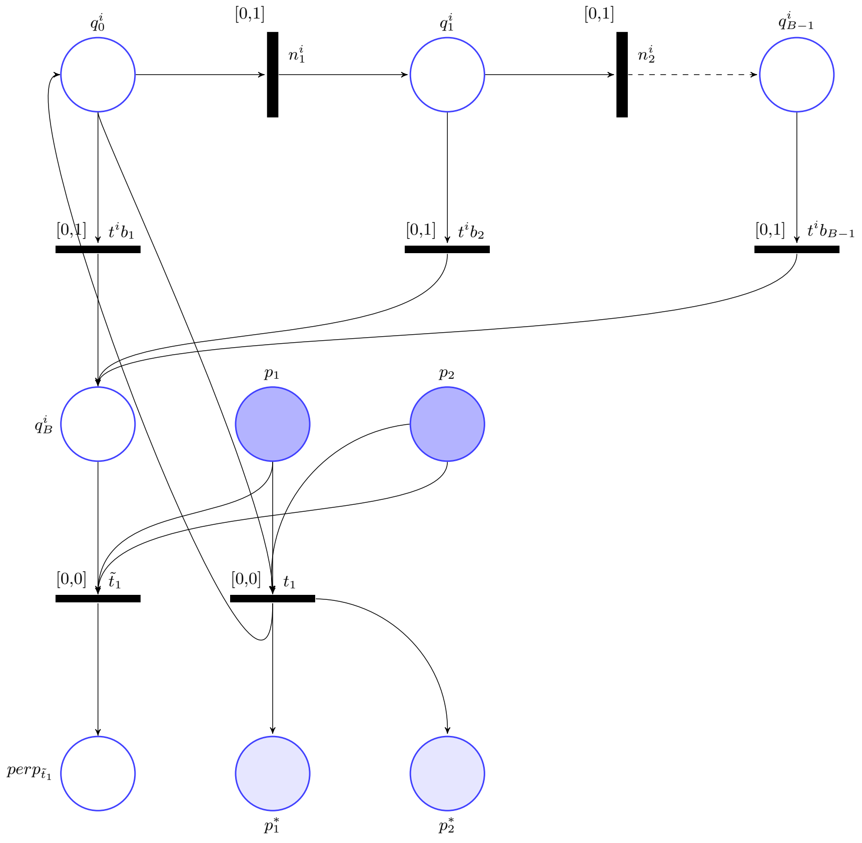

Fig. 4. The Petri net for a local time membrane system with the weak semantics

As a continuation of Figure 4 we show the role q_B^i plays in deciding whether the simulation of the computational step can be continued or not. When the Petri net finishes the simulation of a rule application phase it must check whether the rules corresponding to the chosen transitions form a maximal parallel set of rules. If at the end of the rule application phase at least one rule remains that could be applied, then our choice is obviously not a maximal parallel set of rules. In order to ensure the correct simulation, in this case the Petri net enters into an infinite loop of transitions when reaching state $perp_{\tilde{t}}$ provided $\tilde{t}$ could be applied. Otherwise, control is given back to the semaphore. Figure 5 details the Petri net which is in fact a sub net of the one in Figure 4.

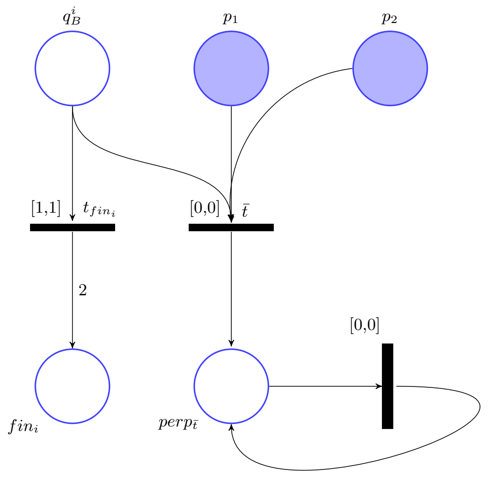

Fig. 5. The Petri net deciding whether a maximal parallel set of rules is reached.

Finally, Figure 6 gives an overall picture of the odd part of the simulating Petri net. The place st_o stimulates the odd phase and tokens are immediately distributed among the places q_0^i, which initiate the simulations of the computations in membranes m_i ($1 \leq i \leq n$). When the computation in m_i is over, in the simulating Petri net the place fin_i obtains a token. If all the places fin_i ($1 \leq i \leq n$) have collected their tokens, then a token is passed over to sem and a new computational phase begins.

We remark that the construction above gives a general method for simulating any time Petri net defined with the weak semantics by a time Petri net understood with the strong semantics.

The case for the membrane system with the strong semantics is possibly a bit simpler, since it is closer to the original semantics of the Petri net model. We define in the next definition the simulating Petri net for a local time membrane system with the strong semantics. The construction is very similar to the ones of the previous subsection, probably it is a little bit easier this time. The only difficulty is to tell when a maximal parallel step is finished. For this purpose, before choosing

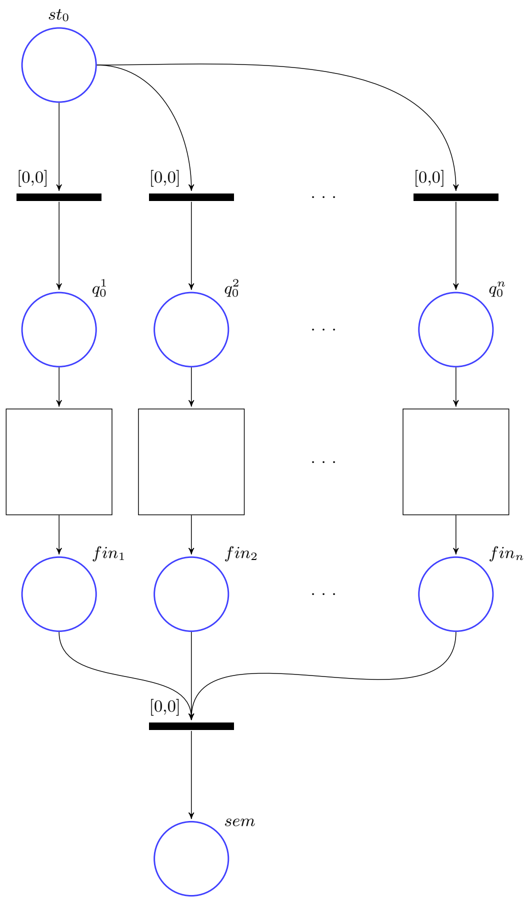

Fig. 6. The overall structure of the Petri net for the weak semantics

the next transition, we implement a test whether there are transitions that could be executed. This involves creating a new copy of the Petri net simulating the left hand sides of the membrane system rules. Moreover, to keep ourselves to the interpretation of the membrane computational step by distinguishing the computational sequences in the different compartments, we assume that there are n sub Petri nets modelling the computations in the different membranes. We detach, as usual, the odd and even phases of the computation. In the odd phase, the token in st_o is immediately distributed to the places st_o^i hence initializing the computation in the sub Petri nets corresponding to membranes m_i.

Definition 18. *Let $\Pi = (V, \mu, w_1, \dots, w_n, R_1, \dots, R_n, \mathcal{I})$ be a local time membrane system with the strong semantics.*

1. *P contains P_0 and P_0^*, where $P_0 = V \times \{1,\ldots,n\}$, $P_0^* = V^* \times \{1,\ldots,n\}$, where $a \in V$ and $(1 \le j \le n)$ are in Definition 11. As usual, P_0 represents the actual configuration of the membrane system. We set $m_0(p) = w_j(a)$ for every place $p = (a,j)$. As before, the places in $V \times \{1,\ldots,n\}$ correspond to the objects on the left hand sides of R_i $(1 \le i \le n)$ of the membrane rules, while the elements of $V^* \times \{1,\ldots,n\}$ correspond to the objects on the right hand sides of the membrane rules labelled by messages. There are auxiliary places, namely those of the semaphore and st_e, st_o^i, cat_i: they govern the simulation of the rule application and communication phases.*
2. *T consists of T_0, T_0^*, $\widetilde{T}_0$ and some auxiliary transitions. As before, let $r_l \in R_i$, where $l \in \{1,\ldots,n_{k_i}\}$. Then let t_l^i denote the transition corresponding to r_l and $T_0 = \{t_l^i \mid 1 \le i \le n, 1 \le l \le k_i\}$. A transition t_l^i connects elements of P_0 to P_0^*: if $p = (a,j)$, then $V(p,t_l^i) = lhs(r_l)(a)$, if $i = j$, and $V(p,t_l^i) = 0$ otherwise. Furthermore, if $p^* = (a_i,j)^*$, then $V(t_l^i,p^*) = rhs(r_l)(a)$, if $i = j$, $V(t_l^i,p^*) = rhs(r_l)(a,out)$, if $i = parent(j)$ and $V(t_l^i,p^*) = rhs(r_l)(a,in_j)$, if $j = parent(i)$ and $V(t_l^i,p^*) = 0$ otherwise. Moreover, (cat_i,t_l^i), (t_l^i,st_o^i), (st_o^i,t_{cat_i}), $(t_{cat_i},cat_i) \in F$.*
   $T_0^* = \{s_j^i \mid 1 \le i \le k, 1 \le j \le n\}$ is defined as before: we let $\{^\bullet(s_j^i)1 \le i \le k, 1 \le j \le n\} \mid\ = P_0^*$, $\{(s_j^i)^\bullet \mid 1 \le i \le k, 1 \le j \le n\} = P_0$ and $V((p_i^*,j),s_j^i) = V(s_j^i,(a_i,j)) = 1$, where $p_i^* = (a_i,j)^*$, and all the other values be 0.
   As to the auxiliary places and transitions, st_o^i and st_e have to check whether there are transitions left to fire. For st_e this is easy: st_e connects to each transition in T_0^*, that is, (st_e,s_l^i), $(s_l^i,st_e) \in F$ and st_e connects to sem by (st_e,tr_e), $(tr_e,sem) \in F$. The places st_o^i need to make a similar check concerning maximal parallel execution: $(st_o^i,\tilde{t}_l^i) \in F$, $(\tilde{t}_l^i,cat_i) \in F$ and $\tilde{t}_l^i$ are such that the connections with the elements of P_0 are the same as in the case of P_0 and T_0 with the same multiplicities, as well. However, $\tilde{t}_l^i$ do not point to P_0^*, they give back all the tokens to P_0 after any of the transitions $\tilde{t}_l^i$ has been fired. When no transition $\tilde{t}_l^i$ is able to fire, st_o^i forwards a token to fin_i and, when each fin_i $(1 \le i \le n)$ possesses a token, they give control back to sem by (fin_i,mon_o), $(mon_o,sem) \in F$. That is, mon_o fires only if every sub Petri net assigned to membrane m_i finishes its computation.
3. *Concerning the intervals: the intervals belonging to the elements of T_0^* are $[0,0]$. If $t_l^i \in T_0$, then $\mathcal{I}(t_l^i) = \mathcal{I}(r_l^i)$. Furthermore, $\mathcal{I}(fin_i) = \mathcal{I}(fin_e) = [1,1]$. All the remaining intervals are $[0,0]$. The semaphore is the same as by the case of the core model.*

We illustrate the sub Petri net corresponding to m_i in Figure 7.

Let $\Pi = (V,\mu,u_1,\ldots,u_n,R_1,\ldots,R_n,\mathcal{I})$ be a local time membrane system. If Π is considered with the weak semantics, then let $N_w(\Pi)$ denote the time Petri net associated to Π according to Definition 17. If Π is understood with the strong semantics, then let $N_s(\Pi)$ denote the Petri net assigned to Π in accordance with Definition 18. Furthermore, if $(w,\mathcal{T})$ is a proper configuration of Π, let $(\nu(w),\nu(\mathcal{T}))$ be the configuration of $N_w(\Pi)$ or of $N_s(\Pi)$, where

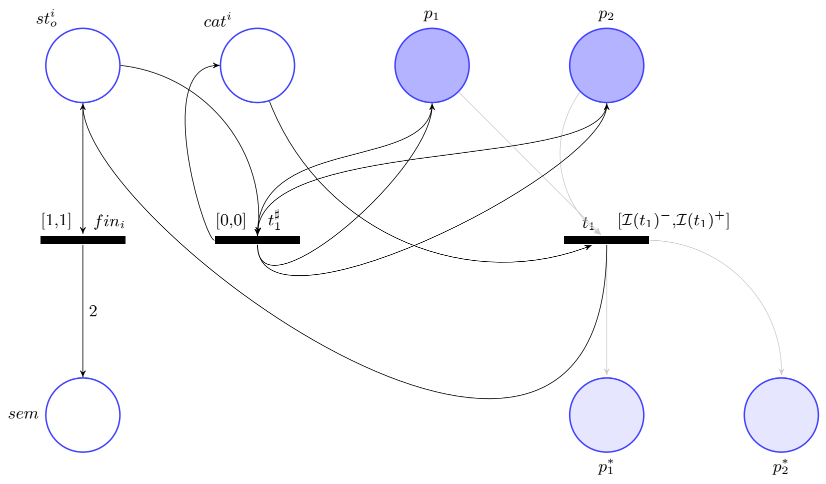

Fig. 7. The Petri net for a local time membrane system with the strong semantics

$\nu(w)(a_i, j) = w(j)(i)$ and $\nu(\mathcal{T})(t_l^i) = \mathcal{T}(r_l)$, where $r_l \in R_i$. In addition, if $p^* = (a_i^*, j)$, let $\nu(w)(p^*) = 0$, as $(w, \mathcal{T})$ is a proper configuration. Moreover, if $W = (w^0, \mathcal{T}^0) \Rightarrow (w^1, \mathcal{T}^1) \Rightarrow \ldots \Rightarrow (w^k, \mathcal{T}^k)$ is a configuration sequence of Π, then $\nu(W) = (\nu(w^0), \nu(\mathcal{T}^0)) \longrightarrow (\nu(w^1), \nu(\mathcal{T}^1)) \longrightarrow \ldots \longrightarrow (\nu(w^k), \nu(\mathcal{T}^k))$ is the corresponding sequence of configurations of $N_w(\Pi)$ or $N_s(\Pi)$. If W is a sequence of proper configurations, then we omit the values $\mathcal{T}_i$ and $\nu(\mathcal{T}_i)$ from the configurations $(w, \mathcal{T})$ and $(\nu(w), \nu(\mathcal{T})_i)$, respectively, since, in this case $\mathcal{T}^j = \mathcal{T}_0$, where $\mathcal{T}_0(r) = 0$ for every $r \in R$.

Theorem 3. *Let $\Pi = (V, \mu, u_1, \ldots, u_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system.*

1. *Let $W = w^0 \Rightarrow w^1 \Rightarrow \ldots \Rightarrow w^k$ be a computational sequence with the weak semantics. Then*
$$w^0 \Rightarrow_W^* w^k \Leftrightarrow N_w(w^0) \longrightarrow_{\nu(W)}^* N_w(w^k).$$
Moreover, $\tau(\nu(W)) = \tau(W) + 3k$, where $\tau(W)$ and $\tau(\nu(W))$ are the total time for W and $\nu(W)$, respectively.
2. *Let $W = w^0 \Rightarrow w^1 \Rightarrow \ldots \Rightarrow w^k$ be a computational sequence with the strong semantics. Then*
$$w^0 \Rightarrow_W^* w^k \Leftrightarrow N_s(w^0) \longrightarrow_{\nu(W)}^* N_s(w^k).$$
Moreover, $\tau(\nu(W)) = \tau(W) + 3k$, where $\tau(W)$ and $\tau(\nu(W))$ are the total time for W and $\nu(W)$, respectively.

Proof. We treat the case of the weak semantics: we give a sketch for the proof of the correctness of our simulation. Assume $\Pi = (V, \mu, u_1, \ldots, u_n, R_1, \ldots, R_n, \mathcal{I})$ is a local time membrane system, let $W = w^0 \Rightarrow w^1 \Rightarrow \ldots \Rightarrow w^k$ be a computational sequence with the weak semantics. We prove the theorem by induction on k.

- $k = 0$: there is nothing to prove.
- $k = l + 1$: Assume we have the statement for l, that is, for the computational sequence $W_l = w^0 \Rightarrow^{\sigma_1} w^1 \Rightarrow^{\sigma_2} \ldots \Rightarrow^{\sigma_l} w^l$ there exists $\xi_l = m^0 \Rightarrow m^1 \Rightarrow \ldots \Rightarrow m^l$ such that $\mu^l = \nu(w^l)$ and $\tau(\nu(\xi_l)) = \tau(W_l) + 3 \cdot l$, where, for any proper configuration w of Π, $\nu(w)$ is defined as above. If q is any place not in P_0, then $\nu(q) = 0$, except for *sem*, where $\nu(w)(sem) = 1$. By the construction of the Petri net $h(t^i_l) = \mathcal{I}(r^i_l)$ also holds. We extend the correspondence ν to the intermediate configurations, as well. The values for the elements of P_0 are defined as before. As to the values of P_0^*: $\nu(w)(a_i^*, j) = w(j)(a_i, here) + w(\mu(j))(a_i, in_j) + \sum_{j=\mu(l)}(a_i, out)$. Let $\sigma_{(i,s)} = \tau_0 r_1 \tau_1 r_2 \ldots r_s \tau_{s+1}$, where $r_1, \ldots, r_s \in R_i$, be a segment of the selection of m_i of length s with the corresponding configurations $((w_1, \mathcal{T}_1), \ldots, (w_s, \mathcal{T}_s))$. Then $\xi_{(i,s)} = \nu(\sigma_{(i,s)}) = \tau_{01} \ldots \tau_{0k_0} t_1 \tau_{11} \ldots \tau_{1k_1} t_2 \ldots t_s \tau_{(s+1)1} \ldots \tau_{(s+1)k_{s+1}}$, where t_j correspond to r_j in $N_w(\Pi)$ and $\tau_j = \tau_{j1} + \ldots + \tau_{jk_j}$, is a computation in $N_w(\Pi)$. Assume $((m_1, h_1), \ldots, (m_s, h_s))$ is the sequence of states corresponding to $\xi_{i,s}$, then we claim that $\nu(w_1), \ldots, \nu(w_s)$ define exactly the same t-markings. For the correspondence of the configurations and t-markings it is enough to prove that, if r can be executed, then t is ready to fire provided t is assigned to r. By Fact 2, it is enough to observe that $\mathcal{T}_i(r) = j + a$ for some $j \in \mathbb{N}$ such that $r \in R^i_j$ and $0 \leq a \leq 1$, and $i_t^- \leq a \leq i_t^+$, where $[i_t^-, i_t^+]$ is the interval corresponding to t as the image of $r \in R^i_j$ in accordance with Definition 17. The statement can be proved by examining the various cases for the next step in $\sigma_{i,s}$.

□

The converse of the theorem holds, too. We state it in a proposition.

Proposition 1. *Let $N = (P, T, F, V, m_0, I)$ be a time Petri net. Then the following statements are valid.*

1. *Let $\Pi = (V, \mu, u_1, \ldots, u_n, R_1, \ldots, R_n, \mathcal{I})$ be be a local time membrane system, assume $N = N_w(\Pi)$. Then, for any proper computational sequence ξ of N, there exists computational sequence W of Π with respect to the weak semantics such that ξ provides exactly the same output as W. Moreover, $\tau(\xi) = \tau(W) + 3k$, where $\tau(W)$ and $\tau(\xi)$ are the total time for W and ξ, respectively.*
2. *Let $\Pi = (V, \mu, u_1, \ldots, u_n, R_1, \ldots, R_n, \mathcal{I})$ be be a local time membrane system, assume $N = N_w(\Pi)$. Then, for any proper computational sequence ξ of N, there exists computational sequence W of Π with respect to the strong semantics such that ξ provides exactly the same output as W. Moreover, $\tau(\xi) = \tau(W) + 3k$, where $\tau(W)$ and $\tau(\xi)$ are the total time for W and ξ, respectively.*

Remark 2. We remark that local time weak semantics does not appear to add anything to the computational power of the membrane system, however local time with the strong semantics seems to increase the computational strength of the P system. We conjecture that, by a modification of a proof of Păun [10] showing that membrane systems with catalytic rules together with priority define recursively enumerable sets of numbers even with two membranes, it might not be difficult to prove that local time membrane systems with catalytic rules and with the strong semantics define recursively enumerable sets of numbers with two membranes. It is not clear to us, however, how the exact strength of a local time membrane system with the strong semantics could be depicted.

7 Applications

The constructions of the previous section makes us possible to apply the results elaborated for time Petri nets for the case of local time membrane systems.

Notation 4 *Let $\Pi = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system. We apply the notation $\Pi = \Pi_\star$, where $\star \in \{w, s\}$ stands for either the weak semantics or the strong semantics, respectively.*

Definition 19. *Let $\Pi_\star = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system.*

1. *A (possibly intermediate) configuration $(w, \mathcal{T})$ is integer valued, if $\mathcal{T}(r) \in \mathbb{N}$ for every $r \in R$.*
2. *Let m_i be a membrane of $\Pi_\star$ for some $1 \leq i \leq n$. Let σ_i be a run for m_i. Then σ_i is integer valued if all of its intermediate configurations are integer valued.*
3. *Let $W = w_0 \Rightarrow \ldots \Rightarrow w_k$ be a computational sequence for $\Pi_\star$. Then W is integer valued, if, for every w_i, every run of w_i is integer valued.*

Observe that, given a membrane system $\Pi_\star$ and a computational sequence W, the condition that W is integer valued is equivalent to the requirement that all the time elapses in every run of W are integers.

Proposition 2. *Let $\Pi_\star = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system and let w' be a configuration reachable from w. Then w' is integer reachable from w.*

Proof. Follows from the main theorem together with the corresponding theorem for time Petri nets presented by Popova-Zeugmann ([13], [14]). □

Definition 20. *Let $\Pi_\star = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system. Then $\Pi_\star$ is bounded, if there exists $K > 0$ such that, for every configuration w of $\Pi_\star$, $|w(j)| < K$ $(1 \leq j \leq n)$. In other words, K is an upper bound for the number of elements in every compartment with regard to any configuration w.*

Proposition 3. *Let $\Pi_\star = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system, assume $\Pi_\star$ is bounded. Then the reachability problem for $\Pi_\star$ is decidable.*

Proof. Follows from Proposition 2. □

Proposition 4. *Let $\Pi_\star = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \mathcal{I})$ be a local time membrane system, assume w' is reachable from w. Then the minimum and maximum distances between w and w' are integers.*

8 Conclusions

In this paper we examined the connections between two models of computations, namely, the membrane systems introduced by Păun [9] and time Petri nets defined along the lines of the papers of Popova-Zeugmann ([12],[13]). First of all, we presented a simulation of membrane systems without dissolution by time Petri nets. The novelty in our result is the fact that the simulating Petri nets manage to retain the locality of firing: transitions can be fired one after the other without structural control, like maximal parallelism, imposed on the order of their execution. Next, we defined local time membrane systems on the analogy of time Petri nets, equipping the computational model with two types of semantics. We showed that both kinds of local time membrane systems can be simulated by time Petri nets with the strong semantics. Finally, we mentioned some statements concerning local time membrane systems that are straightforward consequences of the similar results for time Petri nets by reason of the simulations.

References

1. B. Aman, G. Ciobanu, Adding Lifetime to Objects and Membranes in P Systems. *International Journal of Computers, Communications and Control*, 5(3) (2010) 268–279.
2. B. Aman, G. Ciobanu, Verification of membrane systems with delays via Petri nets with delays. *Theoretical Computer Science*, 598(C) (2015) 87–101.
3. M. Cavaliere, D. Sburlan, Time and Synchronization in Membrane Systems. *Fundamenta Informaticae*, 64(1) (2005) 65–77.
4. M. Cavaliere, D. Sburlan, Timeindependent P Systems Towards a Petri Net Semantics for Membrane Systems. *Lecture Notes in Computer Science*, volume 3365, International Workshop on Membrane Computing, WMC 2004, 239–258, Springer Verlag, Berlin, 2005.
5. M. Ionescu, Gh. Păun, T. Yokomori. Spiking Neural P Systems. *Fundamenta Informaticae*, 71 (2006) 279–308.
6. J. H. C. M. Kleijn and M. Koutny and G. Rozenberg, Towards a Petri Net Semantics for Membrane Systems. *Lecture Notes in Computer Science*, volume 3850, International Workshop on Membrane Computing, WMC 2005, 292–309, Springer Verlag, Berlin, 2005.

7. C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodríguez-Patón. Tissue P Systems. *Theoretical Computer Science*, 296 (2003) 295–326.
8. P. M. Merlin, A Study of the Recoverability of Computing Systems. *PhD thesis*, University of California, Irvine, CA, 1974
9. G. Păun, Computing with Membranes. *Journal of Computer and System Sciences*, 61(1) (2000) 108–143.
10. G. Păun, *Membrane Computing - An Introduction*, Springer Verlag, Berlin, 2002.
11. C. A. Petri, Kommunikation mit Automaten. *Dissertation*, Universität Hamburg, Hamburg, 1962.
12. L. Popova, On time Petri nets. *Journal of Information Processing and Cybernetics*, EIK, 27(4) (1991) 227–244.
13. L. Popova-Zeugmann, Essential States in time Petri nets, *Informatik- Berichte der HUB*, Nr. 96, 1998.
14. L. Popova-Zeugmann, *Time and Petri Nets*, Springer Verlag, Berlin, 2013.