Flattening and Simulation of Asynchronous Divisionless P Systems with Active Membranes

Alberto Leporati[1], Luca Manzoni[2], and Antonio E. Porreca[1]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
{leporati,porreca}@disco.unimib.it
[2] Laboratoire i3S, Université Nice Sophia Antipolis,
CS 40121 – 06903 Sophia Antipolis CEDEX, France
luca.manzoni@i3s.unice.fr

Abstract. We prove that asynchronous P systems with active membranes without division rules can be simulated by single-membrane transition P systems using cooperative rules, even if the synchronisation mechanisms provided by electrical charges and membrane dissolution are exploited. In turn, the latter systems can be simulated by means of place/transition Petri nets, and hence all these models are computationally weaker than Turing machines.

1 Introduction

P systems with active membranes [10] are parallel computation devices inspired by the structure and functioning of biological cells. A tree-like hierarchical structure of membranes divides the space into regions, where *multisets* of objects (representing chemical substances) are located. The systems evolve by means of rules rewriting or moving objects, and possibly changing the membrane structure itself (by dissolving or dividing membranes) or the state of the membranes (by changing their electrical charge).

Under the *maximally parallel* updating policy, whereby all components of the system that can evolve concurrently during a given computation step are required to do so, these devices are known to be computationally universal. Alternative updating policies have also been investigated. In particular, *asynchronous* P systems with active membranes [7], where any, not necessarily maximal, number of non-conflicting rules may be applied in each computation step, have been proved able to simulate partially blind register machines [8], computation devices equivalent under certain acceptance conditions to place/transition Petri nets and vector addition systems [11]. This simulation only requires object evolution (rewriting) rules and communication rules (moving objects between regions).

In an effort to further characterise the effect of asynchronicity on the computational power of P systems, we prove that asynchronous P systems with active membranes without dissolution can be flattened if we allow the use of cooperative

rules, obtaining a system that can be easily simulated by place/transition Petri nets, and as such they are not computationally equivalent to Turing machines: indeed, the reachability of configurations and the deadlock-freeness (i.e., the halting problem) of Petri nets are decidable [2]. This holds even when membrane dissolution, which provides an additional synchronisation mechanism (besides electrical charges) whereby all objects are released simultaneously from the dissolving membrane, is employed by the P system being simulated. Unfortunately, this result does not seem to immediately imply the equivalence with partially blind register machines, as the notion of acceptance for Petri nets employed here is by halting and not by placing a token into a "final" place [8].

The paper is organised as follows: in Section 2 we recall the relevant definitions of (divisionless) P systems with active membranes and place/transition Petri nets; in Section 3 we prove that asynchronous P systems with active membranes are computationally equivalent to their *sequential* version, where a single rule is applied during each computation step; in Section 4 we show that sequential P systems with dissolution rules can be simulated by sequential transition P systems with cooperative rules having only one membrane; finally, in Section 5 we show how sequential single-membrane transition P systems using cooperative rules can be simulated by Petri nets. Section 6 contains our conclusions and some open problems.

2 Definitions

We first recall the definition of P systems with active membranes and its various operating modes.

Definition 1. *A* P system with active membranes *of initial degree* $d \geq 1$ *is a tuple* $\Pi = (\Gamma, \Lambda, \mu, w_{h_1}, \ldots, w_{h_d}, R)$*, where:*

- Γ *is an alphabet, i.e., a finite nonempty set of* objects*;*
- Λ *is a finite set of labels for the membranes;*
- μ *is a membrane structure (i.e., a rooted* unordered *tree) consisting of* d *membranes injectively labelled by elements of* Λ*;*
- $w_{h_1}, \ldots, w_{h_d}$*, with* $h_1, \ldots, h_d \in \Lambda$*, are strings over* Γ*, describing the initial multisets of objects located in the* d *regions of* μ*;*
- R *is a finite set of* rules*.*

Each membrane possesses, besides its label and position in μ, another attribute called *electrical charge*, which can be either neutral (0), positive (+) or negative (−) and is always neutral before the beginning of the computation.

The following four kinds of rules are employed in this paper.

- *Object evolution rules*, of the form $[a \to w]_h^\alpha$
  They can be applied inside a membrane labeled by h, having charge α and containing an occurrence of the object a; the object a is rewritten into the multiset w (i.e., a is removed from the multiset in h and replaced by every object in w).

– *Send-in communication rules*, of the form $a\,[\,]_h^\alpha \to [b]_h^\beta$
  They can be applied to a membrane labeled by h, having charge α and such that the external region contains an occurrence of the object a; the object a is sent into h becoming b and, simultaneously, the charge of h is changed to β.
– *Send-out communication rules*, of the form $[a]_h^\alpha \to [\,]_h^\beta\, b$
  They can be applied to a membrane labeled by h, having charge α and containing an occurrence of the object a; the object a is sent out from h to the outside region becoming b and, simultaneously, the charge of h is changed to β.
– *Dissolution rules*, of the form $[a]_h^\alpha \to b$
  They can be applied to a membrane labeled by h, having charge α and containing an occurrence of the object a; the membrane h is dissolved and its contents are released in the surrounding region unaltered, except that an occurrence of a becomes b.

We recall that the most general form of P systems with active membranes [10] also includes *membrane division rules*, which duplicate a membrane and its contents; however, these rules are not used in this paper.

Each instantaneous configuration of a P system with active membranes is described by the current membrane structure, including the electrical charges, together with the multisets located in the corresponding regions. A computation step changes the current configuration according to the following set of principles:

– Each object and membrane can be subject to at most one rule per step, except for object evolution rules (inside each membrane several evolution rules having the same left-hand side, or the same evolution rule can be applied simultaneously; this includes the application of the same rule with multiplicity).
– When several conflicting rules can be applied at the same time, a nondeterministic choice is performed; this implies that, in general, multiple possible configurations can be reached after a computation step.
– In each computation step, all the chosen rules are applied simultaneously (in an atomic way). However, in order to clarify the operational semantics, each computation step is conventionally described as a sequence of microsteps as follows. First, all evolution rules are applied inside the elementary membranes, followed by all communication and dissolution rules involving the membranes themselves; this process is then repeated to the membranes containing them, and so on towards the root (outermost membrane). In other words, the membranes evolve only after their internal configuration has been updated. For instance, before a membrane dissolution occurs, all chosen object evolution rules must be applied inside it; this way, the objects that are released outside during the dissolution are already the final ones.
– The outermost membrane cannot be dissolved, and any object sent out from it cannot re-enter the system again.

In the *maximally parallel* mode, the multiset of rules to be applied at each step must be maximal, in the sense that no further rule can be added without creating

conflicts. In the *asynchronous* mode, any nonempty multiset of applicable rules can be chosen. Finally, in the *sequential* mode, exactly one rule per computation step is applied. In the following, only the latter two modes will be considered.

A *halting computation* of the P system Π is a finite sequence of configurations $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_n)$, where $\mathcal{C}_0$ is the initial configuration, every $\mathcal{C}_{i+1}$ is reachable from $\mathcal{C}_i$ via a single computation step, and no rule can be applied in $\mathcal{C}_n$. A *non-halting* computation $\boldsymbol{\mathcal{C}} = (\mathcal{C}_i : i \in \mathbb{N})$ consists of infinitely many configurations, again starting from the initial one and generated by successive computation steps, where the applicable rules are never exhausted.

The other model of computation we will employ is Petri nets. In particular, with this term we denote place/transition Petri nets with weighted arcs, self-loops and places of unbounded capacity [4]. A Petri net N is a triple (P, T, F) where P is the set of *places*, T the set of *transitions* (disjoint from P) and $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*. The arcs are weighted by a function $w \colon F \to (\mathbb{N} - \{0\})$. A *marking* (i.e., a configuration) is a function $M \colon P \to \mathbb{N}$. Given two markings M, M' of N and a transition $t \in T$ we say that M' is reachable from M via the firing of t, in symbols $M \to_t M'$, if and only if:

- for all places $p \in P$, if $(p,t) \in F$ and $(t,p) \notin F$ then $M(p) \geq w(p,t)$ and $M'(p) = M(p) - w(p,t)$;
- for all $p \in P$, if $(t,p) \in F$ and $(p,t) \notin F$ then $M'(p) = M(p) + w(t,p)$;
- for all $p \in P$, if both $(p,t) \in F$ and $(t,p) \in F$ then $M(p) \geq w(p,t)$ and $M'(p) = M(p) - w(p,t) + w(t,p)$.

Petri nets are nondeterministic devices, hence multiple markings may be reachable from a given configuration. We call *halting computation* a sequence of markings $(M_0, \ldots M_n)$ where $M_0 \to_{t_1} M_1 \to_{t_2} \cdots \to_{t_n} M_n$ for some $t_1, \ldots, t_n$, and no transition may fire in M_n. Several problems related to the reachability of markings and halting configurations (or *deadlocks*) are decidable [2].

3 Asynchronicity and Sequentiality

In this section we show how it is possible to construct, for every asynchronous P system with active membranes, a *sequential* version that is equivalent to the original one, in the sense that each asynchronous step where more than one rule is applied can be substituted by a sequence of asynchronous steps where the rules are reordered and applied one at a time.

Proposition 1. *Let Π be a P system with active membranes using object evolution, communication, and dissolution rules. Then, the asynchronous and the sequential updating policies of Π are equivalent in the following sense: for each asynchronous (resp., sequential) computation step $\mathcal{C} \to \mathcal{D}$ there exists a series of sequential (resp., asynchronous) steps $\mathcal{C} = \mathcal{C}_0 \to \cdots \to \mathcal{C}_n = \mathcal{D}$ for some $n \in \mathbb{N}$.*

Proof. Every asynchronous computation step $\mathcal{C} \to \mathcal{D}$ consists in the application of a finite multiset of rules $\{e_1, \ldots, e_p, c_1, \ldots, c_q, d_1, \ldots, d_r\}$, where $e_1, \ldots, e_p$

are object evolution rules, $c_1, \ldots, c_q$ are communication rules (either send-in or send-out), and $d_1, \ldots, d_r$ are dissolution rules.

Since evolution rules do not change any charge nor the membrane structure itself, the computation step $\mathcal{C} \to \mathcal{D}$ can be decomposed into two asynchronous computation steps $\mathcal{C} \to \mathcal{E} \to \mathcal{D}$, where the step $\mathcal{C} \to \mathcal{E}$ consists in the application of the evolution rules $\{e_1, \ldots, e_p\}$, and the step $\mathcal{E} \to \mathcal{D}$ in the application of the remaining rules $\{c_1, \ldots, c_q, d_1, \ldots, d_r\}$. Notice that in $\mathcal{E}$ there still exist enough objects to apply these communication and dissolution rules, since by hypothesis $\mathcal{C} \to \mathcal{D}$ is a valid computation step.

Furthermore, notice how there is no conflict between object evolution rules (once they have been assigned to the objects they transform). Therefore, the application of the rules $\{e_1, \ldots, e_p\}$ can be implemented as a series of sequential steps $\mathcal{C} = \mathcal{C}_0 \to \cdots \to \mathcal{C}_p = \mathcal{E}$.

Each membrane can be subject to at most a single rule of communication or dissolution type in the computation step $\mathcal{C} \to \mathcal{D}$; hence, applying one of these rules does not interfere with any other. Thus, these rules can also be serialised into sequential computation steps $\mathcal{E} \to \mathcal{C}_{p+1} \to \cdots \to \mathcal{C}_{p+q+r} = \mathcal{D}$. Once again, all rules remain applicable since they were in the original computation step.

By letting $n = p + q + r$, the first half of the proposition follows. The second part is due to the fact that every sequential computation step is already an asynchronous computation step. □

4 Single-Membrane Transition P Systems

In this section we recall the notion of *transition P system*, imposing as an additional constraint that the system has only one membrane. For a description of a general framework in which these systems can be described see [6]. As proved in [5], these systems are not universal; indeed, a simple simulation by means of Petri nets, inspired by [3], is provided in the next section. Our simulation involves a flattening of the membrane structure and the use of cooperative rules; the first simulation of this type was presented in [1] and, in fact, our construction is similar. Unlike that construction, however, the semantics that we use is sequential and we do not include promoters and inhibitors.

Definition 2. *A single-membrane transition P system is a structure*

$$\Pi = (\Gamma, w, R)$$

where Γ is a finite alphabet, w is a multiset of elements representing the initial state of the system, and R is a set of cooperative rules in the form $v \to w$ where v and w are multisets of objects of Γ.

Notice that the definition is a simplified version of the original definition of transition P systems [9], since specifying the membrane structure is not needed. We can now show that single-membrane transition P systems are equivalent to divisionless P systems with active membranes when operating under the sequential semantics.

Let $\Pi = (\Gamma, \Lambda, \mu, w_{h_1}, \ldots, w_{h_d}, R)$ be a P system with active membranes and $\mathcal{C}$ a configuration of Π. The *flattened encoding* of $\mathcal{C}$ is the multiset $E(\mathcal{C})$ over $(\Gamma \cup \{-, 0, +\}) \times \Lambda$ defined as follows:

1. If there are n copies of the object a contained in a membrane h in $\mathcal{C}$, then $E(\mathcal{C})$ contains n copies of the element (a, h).
2. If a membrane h has charge c, then the object (c, h) is in $E(\mathcal{C})$.

It is easy to see that, for a fixed Π, the encoding function is a bijection between the configurations of Π and its image, that is, the function E is invertible. Hence, for any multiset A that is the encoding of some configuration, the decoding is uniquely identified, i.e., for any configuration $\mathcal{C}$, $E^{-1}(E(\mathcal{C})) = \mathcal{C}$.

Proposition 2. *Let $\Pi = (\Gamma, \Lambda, \mu, w_{h_1}, \ldots, w_{h_d}, R)$ be a P system with active membranes working in the sequential mode and using object evolution, communication, and dissolution rules, with initial configuration $\mathcal{C}_0$. Then, there exists a single-membrane transition P system $\Pi' = \big((\Gamma \cup \{-, 0, +\} \cup \{\bullet\}) \times \Lambda, v, R'\big)$, for some initial multiset v, working in the sequential mode, such that:*

(i) If $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m)$ is a halting computation of Π, then there exists a halting computation $\boldsymbol{\mathcal{D}} = (E(\mathcal{C}_0), \mathcal{D}_1, \ldots, \mathcal{D}_n)$ of Π' such that $\mathcal{D}_n$ is the union of $E(\mathcal{C}_m)$ and the set of all the elements in the form $(\bullet, h)$ where h is a membrane that has been dissolved in $\boldsymbol{\mathcal{C}}$.

(ii) If $\boldsymbol{\mathcal{D}} = (E(\mathcal{C}_0), \mathcal{D}_1, \ldots, \mathcal{D}_n)$ is a halting computation of Π', then there exists a halting computation $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m)$ of Π such that $\mathcal{D}_n$ can be written as the union of the set of elements in the form $(\bullet, h)$, where h is a membrane that was dissolved in $\boldsymbol{\mathcal{C}}$, and the set $E(\mathcal{C}_m)$.

(iii) Π admits a non-halting computation $(\mathcal{C}_0, \mathcal{C}_1, \ldots)$ if and only if Π' admits a non-halting computation $(E(\mathcal{C}_0), \mathcal{D}_1, \ldots)$.

Proof. The main idea is to replace every dissolution rule of a membrane h in R with a cooperative rule such that an object in the form $(\bullet, h)$ is generated and all the objects in the form (a, h) are rewritten to (a, h'), where h' is the lowest ancestor of h in μ that has not been dissolved.

Let $[a]^{\alpha}_{h_1} \to b$ be a dissolution rule in R. Then, R' contains the following cooperative rules:

$$(a, h_1)(\alpha, h_1) \to (b, h_1)(\bullet, h_1). \tag{1}$$

The objects that have h_1 as the second component are then rewritten by means of the following rules:

$$(a, h_1)(\bullet, h_1) \to (a, h_2)(\bullet, h_1) \tag{2}$$

where h_2 is the parent membrane of h_1 in μ. Notice that, if $(\bullet, h_2)$ exists, then membrane h_2 has been dissolved during a previous computation step; this means that there exists another rule of type (2) rewriting all the objects having h_2 as the second component. This process continues as long as there are objects with

the label of a dissolved membrane as their second component (excluding the ones having $\bullet$ as the first component).

An object evolution rule $[a \to w]_h^\alpha$ is simulated by the following cooperative rule:

$$(a,h)(\alpha,h) \to (w_1,h)\dots(w_n,h)(\alpha,h). \tag{3}$$

A send-out communication rule $[a]_{h_1}^\alpha \to [\,]_{h_1}^\beta b$ is replaced by the following rules:

$$(a,h_1)(\alpha,h_1) \to (b,h_2)(\beta,h_1) \tag{4}$$

where h_2 is the parent membrane of h_1 in μ. As mentioned before, if $(\bullet,h_2)$ exists, then a rule of type (2) will subsequently rewrite (b,h_2).

Finally, a send-in communication rule $a\,[\,]_{h_1}^\alpha \to [b]_{h_1}^\beta$ is simulated as follows. Let $(h_n,h_{n-1},\dots,h_2,h_1)$ be a sequence of nested membranes surrounding h_1, i.e., a descending path in the membrane tree μ. For every such sequence, we add the following rules to R':

$$(\bullet,h_{n-1})\cdots(\bullet,h_2)(\alpha,h_1)(a,h_n) \to (\bullet,h_{n-1})\cdots(\bullet,h_2)(\beta,h_1)(b,h_1). \tag{5}$$

These rules rewrite the object (a,h_n) into (b,h_1) if in Π all the membranes between h_n and h_1 have been dissolved. Observe that the number of descending paths leading to h_1 is bounded above by the depth of μ.

Notice how every rule of R' is exactly of one type among (1)–(5); in particular, given a rule in R' of type (1), (3), (4), or (5), it always possible to reconstruct the original rule in R.

Each computation step of Π consisting in the application of an evolution or send-in communication rule is simulated by a single computation step of Π' by means of a rule of type (3) or (5), respectively.

The dissolution of a membrane h_1 in Π requires a variable number of steps of Π': first, a rule of type (1) is applied, then each object in the form (a,h_1) must be rewritten, by using rules of type (2), in order to obtain an object in the form (a,h_n), where h_n is the lowest ancestor membrane of h_1 that has not been dissolved in the original system. The exact number of steps depends on the number of objects located inside h_1 and the number of membranes that have been dissolved. The reasoning is analogous for send-out communication rules, simulated by means of rules of type (4) and (2).

Part (i) of the proposition directly follows from the semantics of the above cooperative rules.

Now let $\boldsymbol{\mathcal{D}} = (\mathcal{D}_0 = E(\mathcal{C}_0), \mathcal{D}_1, \dots, \mathcal{D}_n)$ be a halting computation of Π'. Then there exists a sequence of rules $\boldsymbol{r} = (r_1,\dots,r_n)$ in R' such that

$$\mathcal{D}_0 \to_{r_1} \mathcal{D}_1 \to_{r_2} \cdots \to_{r_{n-1}} \mathcal{D}_{n-1} \to_{r_n} \mathcal{D}_n$$

where the notation $\mathcal{X} \to_r \mathcal{Y}$ indicates that configuration $\mathcal{Y}$ is reached from $\mathcal{X}$ by applying the rule r. Let $f\colon \mathbb{N} \to \mathbb{N}$ be defined as

$$f(t) = \big|\{r_i : 1 \le i \le t \text{ and } r_i \text{ is not of type (2)}\}\big|.$$

We claim that there exists a sequence of rules $\boldsymbol{s} = (s_1, \ldots, s_m)$ such that the computation $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_m)$ of Π generated by applying the rules of $\boldsymbol{s}$, i.e.,

$$\mathcal{C}_0 \to_{s_1} \mathcal{C}_1 \to_{s_2} \cdots \to_{s_{m-1}} \mathcal{C}_{m-1} \to_{s_m} \mathcal{C}_m$$

has the following property $P(t)$ for each $t \in \{0, \ldots, n\}$:

> For all $h \in \Lambda$ and $a \in \Gamma$, if (γ, h) with $\gamma \in \{+, 0, -\}$ is in configuration $\mathcal{D}_t$ of Π', then the number of copies of the objects of the form (a, h') with h' any descendant of h in μ, or h itself, is equal to the number of copies of a contained in the membrane substructure rooted in h in $\mathcal{C}_{f(t)}$, and h has the charge γ. If $(\bullet, h)$ is in $\mathcal{D}_t$, then h does not appear in $\mathcal{C}_{f(t)}$ (having been dissolved before).

We prove this property by induction on t. The case $t = 0$ clearly holds, by the definition of the encoding function: $E(\mathcal{C}_{f(0)}) = E(\mathcal{C}_0) = \mathcal{D}_0$, as $f(0) = |\varnothing|$.

Now suppose $P(t)$ holds for some $t < n$. If r_{t+1} is a rule of type (2) then for each object $a \in \Gamma$, the only change in the objects with a as the first component is when the second component h is the label of a membrane that has been dissolved in Π and the objects retain a as the first component while the second one became the label of the parent membrane of h in μ. Furthermore, no symbol in the form (γ, h), where γ is a charge, is rewritten to a different symbol. Since r_{t+1} is of type (2), we have $f(t+1) = f(t)$ hence $\mathcal{C}_{f(t+1)} = \mathcal{C}_{f(t)}$, and property $P(t+1)$ holds.

On the other hand, if r_{t+1} is not of type (2), then $f(t+1) = f(t) + 1$ by definition. Let $s_{f(t)+1} = s_{f(t+1)}$ be the rule corresponding to the cooperative rule r_{t+1} as described above (an object evolution rule if r_{t+1} is of type (3), a dissolution rule if r_{t+1} is of type (1), and so on). Observe that if r_{t+1} is applicable in $\mathcal{D}_t$, then $s_{f(t)+1}$ is applicable in $\mathcal{C}_{f(t)}$ by induction hypothesis:

- if (γ, h) is in $\mathcal{D}_t$ then the membrane h has charge γ in $\mathcal{C}_{f(t)}$;
- if r_{t+1} is of type (1), (3), or (4) and uses an object (a, h) in $\mathcal{D}_t$, then a copy of a appears in membrane h in $\mathcal{C}_{f(t)}$;
- if r_{t+1} is of type (5) and uses an object (a, h) and $(\bullet, h)$ is in $\mathcal{D}_t$, then the object a appears in $\mathcal{C}_{f(t)}$ inside the membrane having the same label as the lowest ancestor of h in the original membrane structure such that (γ, h) with $\gamma \neq \bullet$ is in $\mathcal{D}_t$.

The configuration $\mathcal{C}_{f(t)+1}$ such that $\mathcal{C}_{f(t)} \to_{s_{f(t)+1}} \mathcal{C}_{f(t)+1}$, due to the semantics of the corresponding rules applied by Π and Π', is such that the property $P(t+1)$ holds.

In particular, $P(n)$ holds: configurations $\mathcal{D}_n$ and $\mathcal{C}_{f(n)}$ have the following properties: the encoding $E(\mathcal{C}_{f(n)})$ is contained in $\mathcal{D}_n$ and all other objects not contained in $E(\mathcal{C}_{f(n)})$ are in the form $(\bullet, h)$, where h is the label of a membrane that has been dissolved during the computation. Notice that $\mathcal{C}_{f(n)}$ is a halting configuration, since otherwise any rule applicable from it could be simulated from $\mathcal{D}_n$ as in statement (i). Furthermore, if an object $(\bullet, h)$ is in $\mathcal{D}_n$ then no object in form (a, h) with $a \in \Gamma$ exists, otherwise further rules of type (2) could

be applied, contradicting the hypothesis that $\mathcal{D}_n$ is a halting configuration. For all membranes h in $\mathcal{C}_{f(n)}$ and for all objects $a \in \Gamma$, the number of copies of a that are inside the membrane h in $\mathcal{C}_{f(n)}$ is equal to the number of objects in the form (a, h) in $\mathcal{D}_n$, and statement (ii) follows.

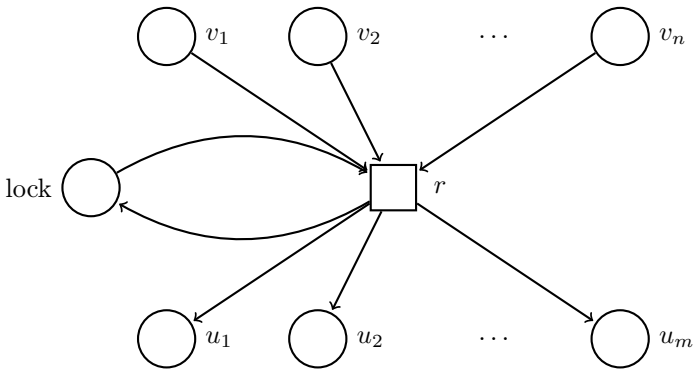
Finally, let us consider a non-halting computation of Π. Each time a computation of Π can be extended by one step by applying a rule, that rule can be simulated by Π' using the same argument employed to prove statement (i), thus yielding a non-halting computation of Π'. Vice versa, in a non-halting computation of Π' it is never the case that infinitely many rules of type (2) are applied sequentially, as only finitely many objects exist at any given time, and eventually they are rewritten to have the form (a, h) without also having the object $(\bullet, h)$. As soon as a rule of type (1), (3), (4), or (5) is applied, the corresponding rule can also be applied by Π, thus yielding a non-halting computation. □

5 Simulation with Petri Nets

The single-membrane transition P systems described in the last section can be simulated by Petri nets in a straightforward way. The idea of using Petri nets as a device for the simulation is originally due to [3].

Proposition 3. *Let $\Pi = (\Gamma, w, R)$ be a single-membrane sequential transition P system. Then, there exists a Petri net N, having Γ among its places, such that $\mathcal{C} \to \mathcal{C}'$ is a computation step of Π if and only if $M \to M'$ is a computation step of N, where $M(a)$ is the number of instances of a in $\mathcal{C}$.*

Proof. The set of places of N is defined as $\Gamma \cup \{\text{lock}\}$, where lock is a place always containing a single token that is employed in order to ensure the firing of at most one transition per step. For each cooperative rule $v_1 \cdots v_n \to u_1 \cdots u_m$ the net has a transition defined as follows:



Notice that the output places need not be distinct, as the multiset in the left hand side may contain multiple occurrences of the same symbol; in that case,

a weighted arc is used. The output places need not be distinct from the input places either; in that case, the net contains a corresponding loop.

The initial marking M_0 of N is given by $M_0(a) = |w|_a$, for all $a \in \Gamma$, where $|w|_a$ is the multiplicity of a in w.

Notice that a transition r in N is enabled exactly when the corresponding rule $r \in R$ is applicable, producing a transition $M \rightarrow_r M'$ corresponding to a computation step $\mathcal{C} \rightarrow_r \mathcal{C}'$ of Π as required. In every moment the number of tokens in a place is equal to the multiplicity of the corresponding object in the configuration of Π. □

By combining Propositions 1, 2, and 3, we can finally prove the following theorem.

Theorem 1. *For every asynchronous P system with active membranes Π using evolution, communication, and dissolution rules, there exists a Petri net N such that (i) every halting configuration of Π corresponds to a halting configuration of N and vice versa (under the encoding of Propositions 2 and 3), and (ii) every non-halting computation of Π corresponds to a non-halting computation of N and vice versa.* □

Notice that, given the strict correspondence of computations and their halting configurations (if any) between the two devices, this result holds both for P systems computing functions over multisets/Parikh vectors and those recognising or generating families of multisets/Parikh vectors, since the only difference between these computing modes is the initial configuration and the acceptance condition; these are translated directly into the simulating Petri net.

6 Conclusions

We have proved that asynchronous P systems with active membranes (without division rules) can be flattened and simulated by single-membrane transition P systems using cooperative rules. These systems can, in turn, be easily simulated by place/transition Petri nets, and hence are not computationally universal. In order to achieve this result, we exploited the equivalence between the asynchronous and the sequential parallelism policies for divisionless P systems with active membranes.

The conjectured equivalence of asynchronous P systems with active membranes and Petri nets does not seem to follow immediately from our result and the previous simulation of partially blind register machines by means of asynchronous P systems with active membranes [7]. Indeed, an explicit signalling (putting a token into a specified place) instead of accepting by halting seems to be required in order to simulate Petri nets with partially blind register machines [8]. Directly simulating Petri nets with asynchronous P systems with active membranes is also nontrivial, since transitions provide a stronger synchronisation mechanism than the limited context-sensitivity of the rules of a P system with active membranes. This equivalence is thus left as an open problem.

Acknowledgements

We would like to thank Luca Bernardinello for his advice on the theory of Petri nets. We would also like to thank the anonymous reviewers for pointing out relevant literature that allowed a simplification of the original construction.

This research was partially funded by Lombardy Region under project NEDD and by the French National Research Agency project EMC (ANR-09-BLAN-0164).

References

1. Agrigoroaiei, O., Ciobanu, G.: Flattening the transition P systems with dissolution. In: Conference on Membrane Computing, CMC 11. LNCS, vol. 6501, pp. 53–64. Springer (2011)
2. Cheng, A., Esparza, J., Palsberg, J.: Complexity results for 1-safe nets. Theoretical Computer Science 147, 117–136 (1995)
3. Dal Zilio, S., Formenti, E.: On the dynamics of PB systems: A Petri net view. In: Workshop on Membrane Computing, WMC3. pp. 153–167. LNCS, Springer (2004)
4. Desel, J., Reisig, W.: Place/transition Petri nets. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri nets I: Basic models, Advances in Petri Nets, vol. 1491, pp. 122–173. Springer (1998)
5. Freund, R.: Asynchronous P systems and P systems working in the sequential mode. In: Workshop on Membrane Computing, WMC4. LNCS, vol. 3365, pp. 36–62. Springer (2005)
6. Freund, R., Verlan, S.: A formal framework for static (tissue) P systems. In: Workshop on Membrane Computing, WMC8. LNCS, vol. 4860, pp. 271–284. Springer (2007)
7. Frisco, P., Govan, G., Leporati, A.: Asynchronous P systems with active membranes. Theoretical Computer Science 429, 74–86 (2012)
8. Greibach, S.A.: Remarks on blind and partially blind one-way multicounter machines. Theoretical Computer Science 7, 311–324 (1978)
9. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000)
10. Păun, Gh.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics 6(1), 75–90 (2001)
11. Peterson, J.L.: Petri net theory and the modeling of systems. Prentice-Hall (1981)