Turing Incompleteness of Asynchronous P Systems with Active Membranes

Alberto Leporati[1], Luca Manzoni[1,2], 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,luca.manzoni,porreca}@disco.unimib.it
[2] I3S Research Lab.,
University Nice Sophia Antipolis,
CS 40121 – 06903 Sophia Antipolis CEDEX, France

Summary. We prove that asynchronous P systems with active membranes without division rules can be simulated by place/transition Petri nets, and hence are computationally weaker than Turing machines. This result holds even if the synchronisation mechanisms provided by electrical charges and membrane dissolution are exploited.

1 Introduction

P systems with active membranes [5] 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 system evolves by means of rules rewriting or moving objects, and possibly changing the membrane structure itself, by dissolving or dividing membranes.

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 [3], 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 [4], computation devices equivalent under certain acceptance conditions to place/transition Petri nets and vector addition systems [6]. 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 can be 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 [1]. 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 [4].

The paper is organised as follows: in Section 2 we recall the relevant definitions; in Section 3 we prove that asynchronous P systems are computationally equivalent to *sequential* P systems, where a single rule is applied during each computation step; in Section 4 we show that dissolution rules in sequential P systems can be replaced by a form of generalised communication rule; finally, in Section 5 we show how P systems using generalised communication rules can be simulated by Petri nets, thus proving our main result. Section 6 contains our conclusions and open problems.

2 Definitions

We 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.

The most general form of P systems with active membranes [5] also includes *membrane division rules*, which duplicate a membrane and its contents; however, division 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 micro-steps 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 must be maximal (i.e., no further rule can be added without creating conflicts) during each step. 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 [2]. 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 [1].

3 Asynchronicity and Sequentiality

In this section we show how it is possible to find, for every asynchronous P system, a *sequential* P system that is equivalent to the original one in the sense that they both halt on the same inputs and produce the same outputs.

The main idea is 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 $\mathcal{C} \to \mathcal{D}$ we have 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 Generalised Communication Rules

In this section we define a variant of P systems with active membranes that we call *generalised communication P systems*, where every evolution, communication, and dissolution rule is replaced by a generalised communication rule, in which an object can at the same time be rewritten and move between membranes while changing their charges. This requires the introduction of an extra membrane charge. Generalised communication rules are introduced in order to simplify the simulation by means of Petri nets, as shown in the next section.

To maintain uniformity in the structure of the rules, we define the external environment as a membrane having label 0 containing all the other membranes and having constant charge.

Definition 2 (Generalised communication rules). *A* generalised communication rule *is a rule of the form*

$$[\cdots[a[\cdots[\,]_{h_n}^{\alpha_n}\cdots]_{h_{i+1}}^{\alpha_{i+1}}]_{h_i}^{\alpha_i}\cdots]_{h_1}^{\alpha_1} \to [\cdots[w[\cdots[\,]_{h_n}^{\beta_n}\cdots]_{h_{j+1}}^{\beta_{j+1}}]_{h_j}^{\beta_j}\cdots]_{h_1}^{\beta_1}$$

On the left-hand side of the rule we have a path in μ (a sequence of nested membranes) consisting of membranes $h_1, \ldots, h_n$ with charges $\alpha_1, \ldots, \alpha_n$, and a single

object a contained in membrane h_i (for some $1 \leq i \leq n$). On the right-hand side the same path appears, with charges $\beta_1, \ldots, \beta_n$, and a multiset w appears in membrane h_j (for some $1 \leq j \leq n$). The rule can be applied when membranes $h_1, \ldots, h_n$ have charges $\alpha_1, \ldots, \alpha_n$ and a copy of a exists inside h_i; the rule removes that copy of a from h_i, adds w to the region h_j, and changes the charges to $\beta_1, \ldots, \beta_n$.

As a special case, $h_1 = 0$ denotes the external environment of the P system; in that case, the charge of h_1 can never be changed.

Definition 3 (Generalised communication P systems). *A generalised communication P system of degree $d \geq 1$ is a structure*

$$\Pi = (\Gamma, \Lambda, \Psi, \mu, w_{h_1}, \ldots, w_{h_d}, R)$$

where the elements Γ, Λ, μ, $w_{h_1}, \ldots, w_{h_d}$ are the same as in standard P systems with active membranes, Ψ is a finite, nonempty set of electrical charges (replacing the standard set of charges $\{+, 0, -\}$), and R consists only of generalised communication rules.

We can now show that generalised communication P systems are equivalent to standard P systems with active membranes (without division rules) when operating under the sequential semantics.

Proposition 2. *Let $\Pi = (\Gamma, \Lambda, \mu, w_{h_1}, \ldots, w_{h_d}, R)$ be a P system with active membranes working in sequential mode and using object evolution, communication, and dissolution rules. Then, there exists a generalised communication P system with active membranes $\Pi' = \big(\Gamma, \Lambda, \{+, 0, -, \bullet\}, \mu, w_{h_1}, \ldots, w_{h_d}, R'\big)$ working in sequential mode, having the same initial configuration $\mathcal{C}_0$ as Π, and 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 $\boldsymbol{\mathcal{D}} = (\mathcal{C}_0, \mathcal{D}_1, \ldots, \mathcal{D}_n)$ of Π' such that each membrane appearing in both $\mathcal{C}_m$ and $\mathcal{D}_n$ contains the same objects and has the same charge in both configurations; if a membrane has dissolved during the computation $\boldsymbol{\mathcal{C}}$, then the corresponding membrane in $\mathcal{D}_n$ is empty and with charge $\bullet$.

(ii) If $\boldsymbol{\mathcal{D}} = (\mathcal{C}_0, \mathcal{D}_1, \ldots, \mathcal{D}_n)$ 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 each membrane appearing in both $\mathcal{C}_m$ and $\mathcal{D}_n$ contains the same objects and has the same charge in both configurations; if a membrane has charge $\bullet$ in $\mathcal{D}_n$, then the corresponding membrane dissolves during the computation $\boldsymbol{\mathcal{C}}$.

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

Proof. The main idea is to replace every dissolution rule in R with a generalised communication rule setting the charge of the membrane h that would be dissolved with a new charge $\bullet$. After setting the charge of h, the objects inside it must be moved to the nearest surrounding membrane having charge different from $\bullet$, in order to ensure that membranes with the same label contain the same objects in

Π and Π'. Send-in communication rules must also be adapted, since the object to be brought in might be not immediately outside the membrane, as a number of membranes with charge $\bullet$ might be interposed.

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

$$[[a]^{\alpha}_{h_1}]^{\beta}_{h_2} \to [[\,]^{\bullet}_{h_1} b]^{\beta}_{h_2} \qquad \text{for } \beta \in \{+, -, 0, \bullet\}, \tag{1}$$

where h_2 is the parent membrane of h_1 in μ. The remaining objects are sent out from h_1 by means of the following rules:

$$[[a]^{\bullet}_{h_1}]^{\beta}_{h_2} \to [[\,]^{\bullet}_{h_1} a]^{\beta}_{h_2} \qquad \text{for } a \in \Gamma,\ \beta \in \{+, 0, -, \bullet\}. \tag{2}$$

Notice that, if $\beta = \bullet$, then membrane h_2 has been dissolved during a previous computation step; this means that there exists another rule of type (2) sending all the objects out from h_2. Hence, the objects never remain stuck in a membrane with charge $\bullet$, and eventually reach a membrane having a different charge.

An object evolution rule $[a \to w]^{\alpha}_{h}$ is simulated by the following generalised communication rule:

$$[a]^{\alpha}_{h} \to [w]^{\alpha}_{h}. \tag{3}$$

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

$$[[a]^{\alpha}_{h_1}]^{\gamma}_{h_2} \to [[\,]^{\beta}_{h_1} b]^{\gamma}_{h_2} \qquad \text{for } \gamma \in \{+, 0, -, \bullet\}. \tag{4}$$

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

Finally, a send-in communication rule $a\ [\,]^{\alpha}_{h_1} \to [b]^{\beta}_{h_1}$ is simulated as follows. Let $(h_n, h_{n-1}, \ldots, h_2, h_1)$ 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':

$$[a\ [\cdots [[\,]^{\alpha}_{h_1}]^{\bullet}_{h_2} \cdots]^{\bullet}_{h_{n-1}}]^{\gamma}_{h_n} \to [[\cdots [[b]^{\beta}_{h_1}]^{\bullet}_{h_2} \cdots]^{\bullet}_{h_{n-1}}]^{\gamma}_{h_n} \qquad \text{for } \gamma \in \{+, 0, -\}. \tag{5}$$

This rule moves the object a into h_1 from the nearest membrane outside h_1 having charge in $\{+, 0, -\}$, ignoring any interposed membrane with charge $\bullet$ (corresponding to a dissolved membrane in Π). 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 located inside h_1 is

sent out to the nearest membrane surrounding h_1 having a charge different from $\bullet$ by using rules of type (2). The exact number of steps depends on the number of objects located inside h_1 and the number of membranes with charge $\bullet$ they have to traverse. The reasoning is analogous for send-out communication rules, simulated by means of rules of type (4) and (2).

Part (i) of the proposition follows from the semantics of generalised communication rules.

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

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

We claim that there exists a sequence of rules $\mathbf{s} = (s_1, \ldots, s_m)$ such that the computation $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_m)$ of Π generated by applying the rules of $\mathbf{s}$, 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 h has a charge among $\{+, 0, -\}$ in configuration $\mathcal{D}_t$ of Π', then the number of copies of a contained in the membrane substructure rooted in h is equal in $\mathcal{D}_t$ and $\mathcal{C}_{f(t)}$, and h has the same charge in both configurations. If h has charge $\bullet$ in $\mathcal{D}_t$, then it does not appear in $\mathcal{C}_{f(t)}$ (having dissolved before).

We prove this property by induction on t. The case $t = 0$ clearly holds, since Π and Π' have the same initial configuration: $\mathcal{C}_{f(0)} = \mathcal{C}_0 = \mathcal{D}_0$, as $f(0) = |\varnothing|$.

Now suppose $P(t)$ holds for some $t < n$. If r_{t+1} is a rule of type (2) then an object is sent out from a membrane with charge $\bullet$; therefore, for each object $a \in \Gamma$, the number of copies of a does not change from $\mathcal{D}_t$ to $\mathcal{D}_{t+1}$ in any subtree rooted in a membrane having charge in $\{+, 0, -\}$; furthermore, no membrane changes its charge. 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 generalised communication 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:

- identically labelled membranes have the same charge in $\mathcal{D}_t$ and $\mathcal{C}_{f(t)}$;
- if r_{t+1} is of type (1), (3), or (4) and uses an object a located inside a membrane h having charge $+$, 0, or $-$ in $\mathcal{D}_t$, then a copy of a also appears in membrane h in $\mathcal{C}_{f(t)}$ for the membrane substructure property;

- if r_{t+1} is of type (5) and uses an object a located inside a membrane h having charge $\bullet$ in $\mathcal{D}_t$, then the object a appears in $\mathcal{C}_{f(t)}$ inside the membrane having the same label as the nearest membrane outside h in $\mathcal{D}_t$ with charge different from $\bullet$.

The configuration $\mathcal{C}_{f(t)+1}$ such that $\mathcal{C}_{f(t)} \rightarrow_{s_{f(t)+1}} \mathcal{C}_{f(t)+1}$, due to the semantics of the corresponding rules applied by Π and Π', is such that the property $P(t+1)$ holds: objects are moved to identically labelled membranes, charges in $\{+, 0, -\}$ are changed in both systems in the same way, and membranes that are set to $\bullet$ by Π' are dissolved by Π.

In particular, $P(n)$ holds: configurations $\mathcal{D}_n$ and $\mathcal{C}_{f(n)}$ have the following properties: membrane substructures rooted in identically labelled membranes contain the same multisets, identically labelled membranes have the same charge if it is in $\{+, 0, -\}$, and membranes having charge $\bullet$ in $\mathcal{D}_n$ do not appear in $\mathcal{C}_{f(n)}$. 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, membranes having charge $\bullet$ in $\mathcal{D}_n$ are empty, otherwise further rules of type (2) could be applied, contradicting the hypothesis that $\mathcal{D}_n$ is a halting configuration. As a consequence, not just the membrane substructures, but the individual membranes contain the same multisets in $\mathcal{D}_n$ and $\mathcal{C}_{f(n)}$, and statement (ii) follows.

Finally, let us consider a non-halting computations 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 reach a membrane having charge different from $\bullet$. 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 generalised communication P systems we introduced in the last section can be straightforwardly simulated by Petri nets.

Proposition 3. *Let $\Pi = (\Gamma, \Lambda, \Psi, \mu, w_{h_1}, \ldots, w_{h_d}, R)$ be a generalised communication P system working in sequential mode. Then, there exists a Petri net N, having $\big((\Lambda \cup \{0\}) \times \Gamma\big) \cup (\Lambda \times \Psi)$ among its places, such that $\mathcal{C} \rightarrow \mathcal{C}'$ is a computation step of Π if and only if $M \rightarrow M'$ is a computation step of N, where*

- *$M(h, a)$ is the number of instances of a in membrane h in $\mathcal{C}$;*
- *$M(0, a)$ is the number of instances of a in the environment in $\mathcal{C}$;*
- *$M(h, \alpha) = 1$ if h has charge α in $\mathcal{C}$, and $M(h, \alpha) = 0$ otherwise;*
- *$M'(h, a)$ is the number of instances of a in membrane h in $\mathcal{C}'$;*
- *$M'(0, a)$ is the number of instances of a in the environment in $\mathcal{C}'$;*

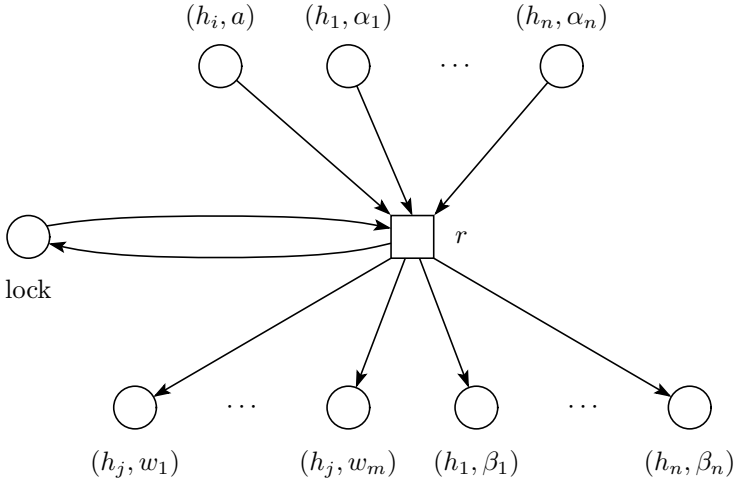
- $M'(h, \alpha) = 1$ *if* h *has charge* α *in* $\mathcal{C}'$*, and* $M'(h, \alpha) = 0$ *otherwise.*

Proof. The set of places of N is defined as $\big((\Lambda \cup \{0\}) \times \Gamma\big) \cup (\Lambda \times \Psi) \cup \{\text{lock}\}$, 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 every generalised communication rule

$$r = [\cdots [a[\cdots [\,]_{h_n}^{\alpha_n} \cdots]_{h_{i+1}}^{\alpha_{i+1}}]_{h_i}^{\alpha_i} \cdots]_{h_1}^{\alpha_1} \to [\cdots [w[\cdots [\,]_{h_n}^{\beta_n} \cdots]_{h_{j+1}}^{\beta_{j+1}}]_{h_j}^{\beta_j} \cdots]_{h_1}^{\beta_1}$$

with $w = w_1 w_2 \cdots w_m$, the net has a transition defined as follows:



Notice that the output places need not be distinct, as the multiset w 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, as w may contain a, or the charge of a membrane may not change. In that case, the net contains a loop.

The initial marking M_0 of N is given by

$$M_0(h, a) = |w_h|_a \qquad M_0(0, a) = 0$$
$$M_0(h, 0) = 1 \qquad M_0(h, \alpha) = 0 \qquad M_0(\text{lock}) = 1$$

for all $h \in \Lambda$, $a \in \Gamma$, and $\alpha \in \Psi - \{0\}$, where $|w_h|_a$ is the multiplicity of a in w_h.

Notice that a transition r in N is enabled exactly when the corresponding rule $r \in R$ is applicable, producing a transition $M \to_r M'$ corresponding to a computation step $\mathcal{C} \to_r \mathcal{C}'$ of Π as required. □

By combining Propositions 1, 2, and 3, we can finally prove our main 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 every halting configuration of* Π *corresponds to a halting configuration of* N

and vice versa (under the encoding of Proposition 3, the charge • denoting dissolved membranes), and 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 (or their Parikh vectors) and those recognising or generating families of multisets (or their Parikh vectors), since the only difference between these various 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 simulated by place/transition Petri nets, and hence are not computationally universal. In order to achieve this result, we proved that the asynchronous and the sequential parallelism policies are equivalent, and that membrane dissolution can be replaced by a generalised form of communication, together with an extra membrane charge.

However, the conjectured equivalence of asynchronous P systems and Petri nets does not seem to follow immediately from our result and the previous simulation of partially blind register machines by means of P systems [3]. 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 [4]. Directly simulating Petri nets with asynchronous P systems 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.

We also conjecture that asynchronous P systems with active membrane remain non-universal even when membrane division rules are allowed. However, if this is the case, a different proof technique than that of Section 3 is required, as our current simulation by Petri nets does not support the creation of new membranes.

Acknowledgements

We would like to thank Luca Bernardinello for his advice on the theory of Petri nets. This research was partially funded by Lombardy Region under project NEDD.

References

1. Cheng, A., Esparza, J., Palsberg, J.: Complexity results for 1-safe nets. Theoretical Computer Science 147, 117–136 (1995)

2. 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)
3. Frisco, P., Govan, G., Leporati, A.: Asynchronous P systems with active membranes. Theoretical Computer Science 429, 74–86 (2012)
4. Greibach, S.A.: Remarks on blind and partially blind one-way multicounter machines. Theoretical Computer Science 7, 311–324 (1978)
5. Păun, Gh.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics 6(1), 75–90 (2001)
6. Peterson, J.L.: Petri net theory and the modeling of systems. Prentice-Hall (1981)