A toolbox for simpler active membrane algorithms

Alberto Leporati, Luca Manzoni, Giancarlo Mauri, Antonio E. Porreca, Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy

Abstract

We show that recogniser P systems with active membranes can be augmented with a priority over their set of rules and any number of membrane charges without loss of generality, as they can be simulated by standard P systems with active membranes, in particular using only *two* charges. Furthermore, we show that more general accepting conditions, such as sending out several, possibly contradictory results and keeping only the first one, or rejecting by halting without output, are also equivalent to the standard accepting conditions. The simulations we propose are always without significant loss of efficiency, and thus the results of this paper can hopefully simplify the design of algorithms for P systems with active membranes.

Keywords: Membrane computing, computational complexity, P system with active membranes

1. Introduction

P systems with active membranes [14] have been extensively investigated as computing devices, both from the computability and the computational complexity standpoints.

By analysing the algorithms for P systems with active membranes described in the literature, it is possible to identify a number of useful and recurring techniques or "design patterns". A standard one is using elementary membrane division to produce all assignments of a set of variables $x_1, \ldots, x_n$ [14]; the results of evaluating a Boolean formula under those assignments can then be combined in several ways:

- by disjunction, allowing the solution of the SAT problem, and thus all **NP**-complete problems [21];
- by counting the number of satisfying assignments against a threshold, allowing the solution of **PP**-complete counting problems [17];
- by alternating disjunctions and conjunctions by means of a tree-shaped membrane structure of depth n, allowing the solution of **PSPACE**-complete problems [20].

Other techniques involve simulating register machines [6] or Turing machines [3], also in their nondeterministic version, by simulating nondeterminism with parallelism as above for solving **NP**-complete problems [10]. Membranes at different nesting levels can also be employed as "subroutines", simulating multiple Turing machines and becoming functionally equivalent to oracles for subproblems [10].

While the main ideas behind those constructions are generally straightforward and show clear affinity with techniques from the theory of traditional computing devices, their implementation unfortunately often involves a number of technical details which obfuscate the big picture. One of the main culprits are the ubiquitous timer objects, which keep the different parts of the P system synchronised and allow the halting of the computation

Email addresses: leporati@disco.unimib.it (Alberto Leporati), luca.manzoni@disco.unimib.it (Luca Manzoni), mauri@disco.unimib.it (Giancarlo Mauri), porreca@disco.unimib.it (Antonio E. Porreca), zandron@disco.unimib.it (Claudio Zandron)

immediately after producing the output, a condition that is usually imposed by the definition of *recogniser P systems* [16] and that often requires extra work to be met.

One of the crucial aspects of the definition of P systems with active membranes is the number of possible membrane charges, which is three in the original definition. Although charges are not needed to solve **PSPACE**-complete problems in polynomial time[1] [4] and two charges suffice to achieve universality [2], having access to a number of charges growing with the size of the input allows a simpler implementation of many algorithms. For instance, when this is allowed, the simulation of bounded-tape Turing machines becomes trivial [10]. In that paper, an arbitrary number of charges was reduced to three without loss of efficiency, but only in a very restrictive set of circumstances (essentially, no communication with adjacent membranes is allowed, and the membranes must behave deterministically).

The purpose of this paper is twofold. On the one hand, we want to understand which features of recogniser P systems with active membranes are actually essential to characterise their behaviour. On the other hand, we want to provide an array of useful extensions which can be added to P systems with active membranes but can be simulated by the original model without loss of efficiency. This will hopefully reduce the amount of "boilerplate code" (repetitive rules unrelated to the main algorithm) in proofs and allow focusing on a higher-level description of P systems, such as dividing membranes working in parallel and their communication patterns.

The formally redundant but convenient features we describe in this paper are the ability to use any number of charges, any partial priority ordering of rules (as in the original definition of transition P systems [13]), and the ability to output the result of the computation in less restrictive ways, such as not requiring the P system to halt after having sent out the result, or rejecting by halting without output. Furthermore, we show that all these enhancements can be simulated efficiently by standard recogniser P systems with active membranes using only *two* charges (even when working in super-polynomial time).

2. Basic notions

For an introduction to membrane computing and the related notions of formal language theory and multiset processing, we refer the reader to *The Oxford Handbook of Membrane Computing* [15] or to any introductory membrane computing survey [18]. Here we recall the formal definition of P systems with active membranes using weak non-elementary division rules [14, 22].

Definition 1. A *P system with active membranes with weak non-elementary division rules* 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 non-empty set of symbols, usually called *objects*;
- Λ is a finite set of labels for the membranes;
- μ is a membrane structure (i.e., a rooted *unordered* tree, usually represented by nested brackets) consisting of d membranes labelled by elements of Λ in a one-to-one way;
- $w_{h_1}, \ldots, w_{h_d}$, with $h_1, \ldots, h_d \in \Lambda$, are multisets (finite sets with multiplicity) of objects in Γ, describing the initial contents of 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 rules in R are of the following types:

[1] However notice that, in the absence of membrane dissolution rules, the lack of charges seems to reduce the efficiency of P systems [9].

(a) *Object evolution rules*, of the form $[a \to w]_h^\alpha$
They can be applied inside a membrane labelled 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 the objects in w).

(b) *Send-in communication rules*, of the form $a\,[\ ]_h^\alpha \to [b]_h^\beta$
They can be applied to a membrane labelled 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.

(c) *Send-out communication rules*, of the form $[a]_h^\alpha \to [\ ]_h^\beta\ b$
They can be applied to a membrane labelled 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 becomes β.

(d) *Dissolution rules*, of the form $[a]_h^\alpha \to b$
They can be applied to a membrane labelled by h, having charge α and containing an occurrence of the object a; the membrane is dissolved and its contents are left in the surrounding region unaltered, except that an occurrence of a becomes b.

(e) *Elementary division rules*, of the form $[a]_h^\alpha \to [b]_h^\beta\ [c]_h^\gamma$
They can be applied to a membrane labelled by h, having charge α, containing an occurrence of the object a but having no other membrane inside (an *elementary membrane*); the membrane is divided into two membranes having label h and charges β and γ; the object a is replaced, respectively, by b and c, while the other objects of the multiset are replicated in both membranes.

(f') *Weak non-elementary division rules, of the form* $[a]_h^\alpha \to [b]_h^\beta\ [c]_h^\gamma$
They can be applied to a membrane labelled by h, having charge α, and containing an occurrence of the object a, even if it contains further membranes; the membrane is divided into two membranes having label h and charges β and γ; the object a is replaced, respectively, by b and c, while the rest of the contents (including whole membrane substructures) is replicated in both membranes.

The instantaneous *configuration* of a membrane consists of its label h, its charge α, and the multiset w of objects it contains at a given time. It is denoted by $[w]_h^\alpha$. The *(full) configuration* C of a P system Π at a given time is a rooted, unordered tree. The root is a node corresponding to the external environment of Π, and has a single subtree corresponding to the current membrane structure of Π. Furthermore, the root is labelled by the multiset located in the environment, and the remaining nodes by the configurations $[w]_h^\alpha$ of the corresponding membranes.

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 can be applied simultaneously.

- The application of rules is *maximally parallel*: each object appearing on the left-hand side of evolution, communication, dissolution or division rules must be subject to exactly one of them (unless the current charge of the membrane prohibits it). Analogously, each membrane can only be subject to one communication, dissolution, or division rule (types (b)–(f')) per computation step; these rules will be called *blocking rules* in the rest of the paper. In other words, the only objects and membranes that do not evolve are those associated with no rule, or only to rules that are not applicable due to the electrical charges.

- 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 whereby each membrane evolves only after their internal configuration (including, recursively, the configurations of the membrane substructures it contains) has been updated. For instance, before a membrane division occurs, all chosen object evolution rules must be applied inside it; this way, the objects that are duplicated during the division are already the final ones.

- The outermost membrane cannot be divided or dissolved, and any object sent out from it cannot re-enter the system again.

A *halting computation* of the P system Π is a finite sequence $\vec{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ of configurations, 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 rules of Π are applicable in $\mathcal{C}_k$. A *non-halting* computation $\vec{\mathcal{C}} = (\mathcal{C}_i : i \in \mathbb{N})$ consists of infinitely many configurations, again starting from the initial one and generated by successive computation steps, where the applicable rules are never exhausted.

P systems can be used as language *recognisers* by employing two distinguished objects yes and no: we assume that all computations are halting, and that either one copy of object yes or one of object no is sent out from the outermost membrane, and only in the last computation step, in order to signal acceptance or rejection, respectively. If all computations starting from the same initial configuration are accepting, or all are rejecting, the P system is said to be *confluent*. If this is not necessarily the case, then we have a *non-confluent* P system, and the overall result is established as for nondeterministic Turing machines: it is acceptance iff an accepting computation exists [16]. Most of the existing literature deals with confluent P systems, although some results of this paper also apply to non-confluent ones.

In order to solve decision problems (or, equivalently, decide languages), we use *families* of recogniser P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$. Each input x is associated with a P system Π_x deciding the membership of x in a language $L \subseteq \Sigma^\star$ by accepting or rejecting. The mapping $x \mapsto \Pi_x$ must be efficiently computable for inputs of any length, as discussed in detail in [11].

Definition 2. A family of P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ is *(polynomial-time) uniform* if the mapping $x \mapsto \Pi_x$ can be computed by two polynomial-time deterministic Turing machines E and F as follows:

- $F(1^n) = \Pi_n$, where n is the length of the input x and Π_n is a common P system for all inputs of length n with a distinguished input membrane.
- $E(x) = w_x$, where w_x is a multiset encoding the specific input x.
- Finally, Π_x is simply Π_n with w_x added to its input membrane.

The family $\mathbf{\Pi}$ is said to be (polynomial-time) semi-uniform if there exists a single deterministic polynomial-time Turing machine H such that $H(x) = \Pi_x$ for each $x \in \Sigma^\star$.

Any explicit encoding of Π_x is allowed as output of the construction, as long as the number of membranes and objects represented by it does not exceed the length of the whole description, and the rules are listed one by one. This restriction is enforced in order to mimic a (hypothetical) realistic process of construction of the P systems, where membranes and objects are presumably placed in a constant amount during each construction step, and require actual physical space proportional to their number; see also [11] for further details on the encoding of P systems.

In this paper we also take advantage of *rule priorities*, as in the original paper introducing P systems [13]. A priority is any partial order $\preceq$ of the set of rules such that, whenever a conflict between rules arises, only those with higher priority can be applied; as usual, when two rules are incomparable with respect to $\preceq$, any conflict is resolved via a nondeterministic choice. Furthermore, we also allow *generalised charges*, that is, any set $\Psi \supseteq \{+, 0, -\}$ of charges may be used [10]. Rule priorities and generalised charges will be proved redundant in Section 4. It was already known that two charges suffice in order to obtain universality [2] and a solution in polynomial time of **NP**-complete problems [1]; the solution proposed here applies to P systems working under any time upper bound, incurring only in a polynomial slowdown.

3. Generalised acceptance conditions

In this section we propose a more flexible variant of recogniser P system, where we do not require that a single output object is sent to the environment at the last step of the computation, or even that the P system halts. This allows to omit of a number of technical details from membrane computing solutions, which are sometimes unrelated to the main algorithm but are still required in order to ensure compliance to the formal definition of

recogniser P systems. We prove that there is no loss of generality in using these variants of accepting condition, as they can always be simulated without significant loss of efficiency by the standard one; we show this result first for P systems with priority and generalised charges, and in a later section for standard P systems with active membranes.

Definition 3. A *generalised recogniser P system* Π is a P system employing two distinguished objects yes and no and behaving in any of the three following ways:

1. It sends out an instance of object yes from its outermost membrane before sending out any instance of object no; it can later send out any combination of objects yes and no, and is not required to halt.
2. It sends out an instance of object no from its outermost membrane before sending out any instance of object yes; it can later send out any combination of objects yes and no, and is not required to halt.
3. It halts without sending out neither an instance of yes, nor an instance of no.

The P system Π is said to *accept* in case 1, and to *reject* in case 2. The behaviour of case 3 can be interpreted as either accepting or rejecting, according to a specified convention.

It is trivial to observe that a standard recogniser P system [16] is a special case of generalised recogniser P system, always halting and sending out exactly an instance of yes or no and only in the last computation step. Furthermore, other acceptance conditions proposed in the literature [12] are also special cases of generalised recogniser P systems; in particular, we have *acknowledger* P systems (which accept by sending out one or more instances of yes and reject by halting without output) and *recogniser*$_{\geq 1}$ P systems (which accept by sending out one or more instances of yes and reject by sending out one or more instances of no). The only notable case not covered by the notion of generalised recogniser P systems is accepting by outputting yes while rejecting by not halting; since P systems are known to be universal [2], this acceptance condition characterises the whole class of recursively enumerable sets.

3.1. Ensuring output on halting

We can now show that standard recogniser P systems with priority and generalised charges solve exactly the same problems as generalised recogniser P systems using the same features with polynomial slowdown. Priority and generalised charges will then be eliminated, also without loss of efficiency, in Section 4. We begin by reducing case 3 of Definition 3 to one of the other two cases: case 2 if halting without output is interpreted as rejecting, or case 1 if it is interpreted as accepting. The idea is to have a timer located inside the outermost membrane, which is sent out as a no object (or as a yes object) if it does not receive a signal for $2d$ consecutive steps, where d is the depth of the membrane structure. This signal indicates that at least one rule was applied in the P system being simulated in the last $2d$ steps, and is propagated as an object ♣ towards the outermost membrane.

Let Π be the generalised recogniser being simulated, and let Π' be the standard recogniser simulating it; both P systems have priority and generalised charges. The initial membrane structure of Π' is identical to that of Π. Inside each membrane, besides the original multiset, we place an instance of $\diamondsuit$ and one of $\heartsuit$; finally, the outermost membrane also contains an instance of the timer object T_d.

The rules of Π are modified in Π' so that their application is always detectable; in order to do so, we always either change the charge of a membrane where a rule was applied to a new, specific charge (for rules involving the membranes themselves) or produce an extra object on the right-hand side (for object evolution rules). Assuming $h \in \Lambda$, $\alpha, \beta, \gamma \in \Psi$, $a, b, c \in \Gamma$, and $w \in \Gamma^\star$, the new rules are determined as follows:

$$[a \to w]_h^\alpha \quad \text{becomes} \quad [a \to w\ \clubsuit]_h^\alpha \tag{1}$$

$$a\ [\ ]_h^\alpha \to [b]_h^\beta \quad \text{becomes} \quad a\ [\ ]_h^\alpha \to [b]_h^{\widetilde{\beta}}$$

$$[a]_h^\alpha \to [\ ]_h^\beta\ b \quad \text{becomes} \quad [a]_h^\alpha \to [\ ]_h^{\widetilde{\beta}}\ b \tag{2}$$

$$[a]_h^\alpha \to b \quad \text{remains identical}$$

$$[a]_h^\alpha \to [b]_h^\beta\ [c]_h^\gamma \quad \text{becomes} \quad [a]_h^\alpha \to [b]_h^{\widetilde{\beta}}\ [c]_h^{\widetilde{\gamma}}$$

Here the new charges of the form $\widetilde{\alpha}$, with $\alpha \in \Psi$, encode the new charge of the membrane and the information that a rule involving that membrane was applied in the previous step. If object evolution rules were applied, then a corresponding number of objects ♣ appear.

In membranes where no blocking rule was applied, the charge is not of the form $\widetilde{\alpha}$. In that case, the following rule (which has lower priority) is applied instead:

$$[\heartsuit]_h^\alpha \to [\ ]_h^{\alpha'} \ \# \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi \tag{3}$$

The object ♡ is restored at each computation step by the following rules:

$$[\diamondsuit \to \diamondsuit\ \heartsuit]_h^\alpha \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$
$$[\diamondsuit \to \diamondsuit\ \heartsuit]_h^{\widetilde{\alpha}} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$
$$[\diamondsuit \to \diamondsuit\ \heartsuit]_h^{\alpha'} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$

(Notice that these rules impede the halting of the P system, but this is allowed by cases 2 and 1 of Definition 3.)

Now each membrane has either a charge of the form $\widetilde{\alpha}$ or one of the form α'. This denotes that we will now perform a signal propagation step, rather than a step simulating rules of Π. All instances of ♣, both those just created by applying rule (1) and those created in previous steps, are propagated one level up (except for the outermost membrane k) and simultaneously change all charges $\widetilde{\alpha}$ and α' to plain α:

$$[\clubsuit]_h^{\widetilde{\alpha}} \to [\ ]_h^\alpha\ \clubsuit \qquad \text{for } h \in \Lambda - \{k\} \text{ and } \alpha \in \Psi \tag{4}$$
$$[\clubsuit]_h^{\alpha'} \to [\ ]_h^\alpha\ \clubsuit \qquad \text{for } h \in \Lambda - \{k\} \text{ and } \alpha \in \Psi \tag{5}$$

The following rules, with priority lower than (4) and (5), create and immediately propagate a new signal object if a rule involving the membrane was applied and no object ♣ was already present:

$$[\heartsuit]_h^{\widetilde{\alpha}} \to [\ ]_h^\alpha\ \clubsuit \qquad \text{for } h \in \Lambda - \{k\} \text{ and } \alpha \in \Psi$$

If a membrane has charge α' (no rule involving that membrane was applied in the previous step) and there is no ♣ to propagate, then ♡ changes the charge to α with the following rules with lower priority:

$$[\heartsuit]_h^{\alpha'} \to [\ ]_h^\alpha\ \# \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$

Any extra occurrence of ♡ is always deleted (i.e., rewritten into the empty multiset) by the following rules, which have minimal priority:

$$[\heartsuit \to \epsilon]_h^\alpha \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$
$$[\heartsuit \to \epsilon]_h^{\widetilde{\alpha}} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$
$$[\heartsuit \to \epsilon]_h^{\alpha'} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$

Any extra occurrence of ♣ is deleted *only in the propagation steps* by the following rules with minimal priority (which are thus only enabled if the signal has already been propagated from the current membrane):

$$[\clubsuit \to \epsilon]_h^{\widetilde{\alpha}} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$
$$[\clubsuit \to \epsilon]_h^{\alpha'} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$

The timer object T_t (with $0 \le t \le d$) in the outermost membrane k counts down in the simulation steps, when the charge of that membrane is one of the original ones:

$$[T_t \to T_{t-1}]_k^\alpha \qquad \text{for } \alpha \in \Psi \text{ and } 0 < t \le d$$

In order to reset the timer when a signal object ♣ reaches the outermost membrane, that object changes the charge, currently of the form $\widetilde{\alpha}$ or α', to a new charge $\alpha_\clubsuit$, whose presence denotes that the charge of the outermost membrane of Π is α and at least one rule was applied in the last d simulated steps:

$$[\clubsuit]_k^{\widetilde{\alpha}} \to [\ ]_k^{\alpha_\clubsuit}\ \# \qquad \text{for } \alpha \in \Psi$$

All original rules related to the outermost membrane, which have a plain charge $\alpha \in \Psi$ on the left-hand side, must be duplicated in order to maintain the same behaviour when the left-hand charge is $\alpha_\clubsuit$; the right-hand side must remain unchanged, i.e., the subscript $\clubsuit$ is dropped when changing the charge to one of the form $\widetilde{\beta}$ or β' in send-out rules (2) and (3).

When the charge of the outermost membrane k has the form $\alpha_\clubsuit$, the counter is reset to d:

$$[\,T_t \to T_d\,]_k^{\alpha_\clubsuit} \qquad \text{for } \alpha \in \Psi \text{ and } 0 \leq t \leq d$$

If, however, the timer reaches 0 while the charge of the outermost membrane k has no subscript $\clubsuit$, this means that no rule of Π was simulated by the P system Π' in the last d steps. We can thus assume that Π has halted, and send out the timer as a no object:

$$[\,T_0\,]_k^{\alpha} \to [\ ]_k^{\widetilde{\alpha}}\ \mathsf{no} \qquad \text{for } \alpha \in \Psi$$

If, on the other hand, halting without output is interpreted as accepting, we send out a yes object instead:

$$[\,T_0\,]_k^{\alpha} \to [\ ]_k^{\widetilde{\alpha}}\ \mathsf{yes} \qquad \text{for } \alpha \in \Psi$$

If no object yes or no has been previously sent out, this no (or yes) object becomes the result of the computation, otherwise it does not change the previous result according to cases 1 and 2 of Definition 3.

The computation time of the P system Π' is as follows: if Π sends out a yes or no object at step t, then the same object is sent out by Π' at step $2t-1$ (the corresponding simulation step); if, on the other hand, Π rejects by halting after t steps without output, then Π' sends out a no object at time $2t-1+2d$, i.e., the time required for simulating the t steps of Π, plus an upper bound on the time needed in order to detect that no signal from any membrane (including the deepest one) has been generated during the currently simulated step, implying that the simulated P system has halted.

An example of the generation of the output no while simulating a P system that rejects by halting without output is depicted in Figure 1.

3.2. Ensuring halting on output

Having reduced case 3 to case 1 or 2 of Definition 3, we still need to ensure that a single output object is sent out of the P system, and only in the last computation step when simulating cases 1 and 2, in order to prove that each generalised recogniser can be replaced by a standard recogniser without significant loss of efficiency. We can further ensure that all membranes of the simulating system have a new, distinguished charge $\spadesuit$ with no associated rules in the last configuration, denoting that the P system is halting; this technical detail will prove useful in Section 4.

Let Π be a generalised recogniser P system which always produces output (i.e., either accepts by case 1 or rejects by case 2) but not necessarily a unique output, and that does not necessarily halt. We design a standard recogniser P system Π' satisfying the requirements above.

The initial configuration of Π' is exactly the same as Π, except that each membrane contains as many instances of a new object $\spadesuit$ as the number of its children membranes, and the outermost membrane contains an instance of a new object R_d. The rules and the alphabet of Π' include all those of Π, except as described below.

The P system Π' executes all rules of Π, with the same priority, except for those sending out the result of the computation from the outermost membrane, while simultaneously doubling the amount of $\spadesuit$ contained inside each membrane by using the following rules, which are only enabled by the original charges of Π:

$$[\spadesuit \to \spadesuit\,\spadesuit]_h^{\alpha} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi \tag{6}$$

Notice that these rules do not compete with the original rules of Π, since they are object evolution rules.

When Π sends out the result object yes or no from the outermost membrane k by means of a rule of the form

$$[a]_k^{\alpha} \to [\ ]_k^{\beta}\ \mathsf{yes} \qquad\qquad [a]_k^{\alpha} \to [\ ]_k^{\beta}\ \mathsf{no}$$

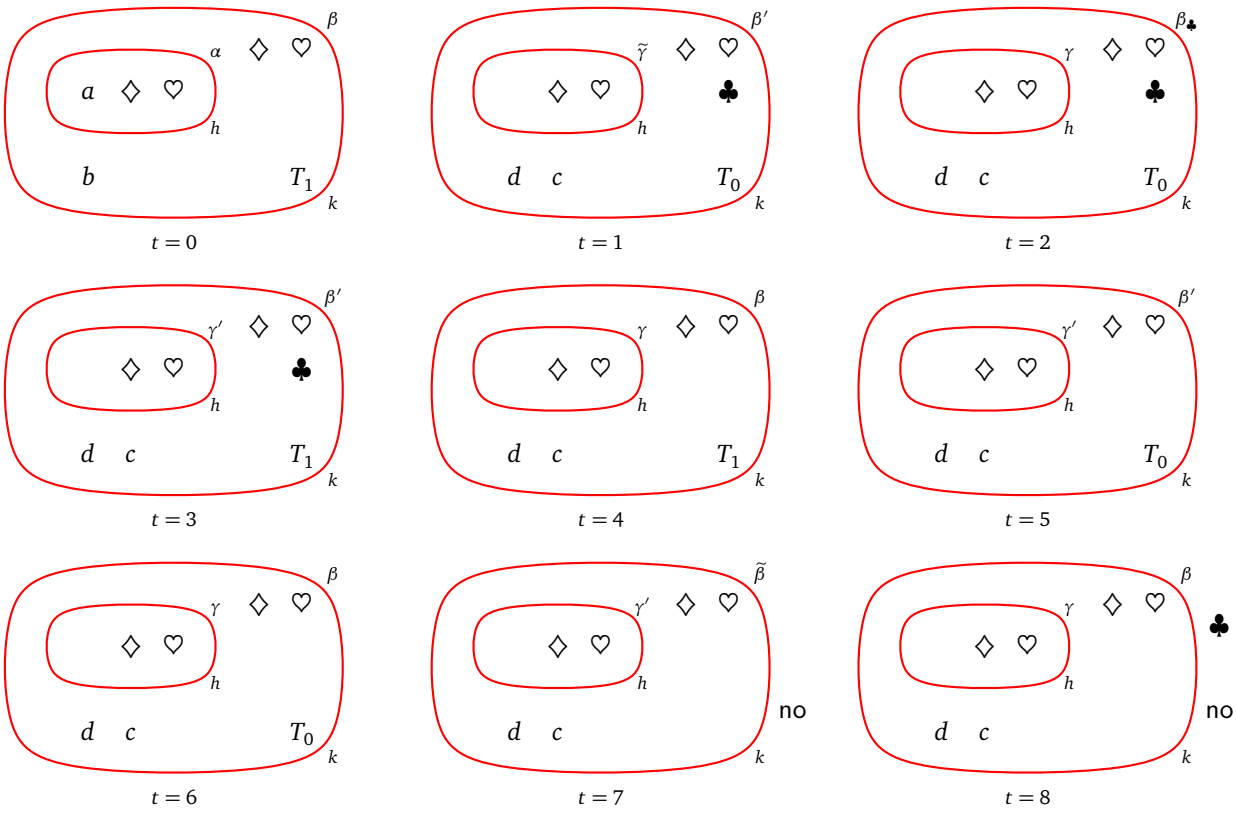

Figure 1: Simulation of a generalised recogniser P system Π that rejects by halting without output. The initial simulated configuration is $[[a]_h^\alpha\ b]_k^\beta$, and the rules of the system are $[a]_h^\alpha \to [\ ]_h^\gamma\ c$ and $[b \to d]_k^\beta$; the P system thus immediately reaches configuration $[[\ ]_h^\gamma\ c\ d]_k^\beta$ and halts. The configuration depicted here at time $t = 0$ encodes the initial configuration of Π. The two original rules are simulated in the computation step leading to the configuration at $t = 1$. Two signal objects ♣ are produced, one by the object evolution rule producing d (visible at time $t = 1$ inside membrane k) and one from within membrane h (appearing at $t = 2$ in k). The first one to appear sets the charge of the outermost membrane k to $\beta_\clubsuit$, signalling that at least one rule was applied in the simulated P system in the last $d = 1$ steps; this restores the timer T_0 to its original value $T_d = T_1$. The configuration at time $t = 2$ encodes the final configuration $[[\ ]_h^\gamma\ c\ d]_k^\beta$ of Π. Since Π now halts, none of its rules are simulated in the following steps, and thus no signal object ♣ is produced. The timer object T_t is decremented every two steps (here at $t = 5$) without being reset, which ultimately leads to T_0 being sent out as no ($t = 7$), thus producing the required output. Notice that the simulating P system does *not* halt, but continues priming and unpriming the charges of the membranes. The junk objects #, which are inert, are not represented in the picture.

the P system Π′ applies instead a rule of the form

$$[a]_k^\alpha \to [\ ]_k^{\mathsf{yes}}\ \#\qquad\qquad [a]_k^\alpha \to [\ ]_k^{\mathsf{no}}\ \# \tag{7}$$

These new rules maintain the same priority as the original ones.

The final result of the computation is thus temporarily stored in the charge of the outermost membrane, and a "junk" object is sent out instead. Notice that, since the charges yes and no are new, the objects of the original alphabet Γ of Π cannot apply any rule inside the outermost membrane. The other membranes might continue computing; we now propagate the information about having produced output towards the internal membranes in order to stop the computation.

Notice that the number of objects ♠ has always been kept at least equal to the number of children membranes during the computation, even when taking membrane division into account (the membranes can at most double in number during each step). When the charge of the outermost membrane of Π′ becomes yes or no, the rules (6) become disabled for the outermost label, and the following rules *with priority lower than* (6) *but higher than the*

simulated rules of Π become now applicable:

$$\spadesuit\,[\ ]_h^\alpha \to [\#]_h^\spadesuit \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi \tag{8}$$

The charge of each children membrane thus changes to $\spadesuit$. Notice that the instances of object $\spadesuit$ in excess of the number of children membrane become inert, since all their rules are now disabled (all reachable membranes now having charge $\spadesuit$). The new charge $\spadesuit$ also disables the rules of type (6) for membrane h, enabling those of type (8) for its children membranes. This propagates the charge $\spadesuit$ to the next level, and so on.

The P system reaches a configuration where all membranes, except the outermost one, have charge $\spadesuit$ exactly d steps after applying one of the rules in (7). The timer R_d inside the outermost membrane k, also enabled when rule (7) is applied, counts these d steps, using the rules

$$[R_i \to R_{i-1}]_k^{\mathsf{yes}} \qquad [R_i \to R_{i-1}]_k^{\mathsf{no}} \qquad \text{for } 0 < i \le d$$

When reaching zero, the object R_0 is finally sent out as the result of the computation while setting the charge of the remaining membrane to $\spadesuit$:

$$[R_0]_k^{\mathsf{yes}} \to [\ ]_k^\spadesuit\ \mathsf{yes} \qquad [R_0]_k^{\mathsf{no}} \to [\ ]_k^\spadesuit\ \mathsf{no}$$

Notice that Π' has exactly the same number of computations as Π and with the same result; indeed, the new rules do not interfere with the simulation of Π while this is still running, and the last phase, where all charges become $\spadesuit$, is deterministic. Furthermore, if Π sends out its first result object at time t, then Π' sends out the same result *and halts* at time $t + d + 1$.

By combining the results of Sections 3.1 and 3.2 we obtain:

Lemma 4. *Let* Π *be a confluent (resp., non-confluent) generalised recogniser P system with priority and generalised charges working in time* t*. Then, there exists a standard confluent (resp., non-confluent) recogniser P system with priority and generalised charges having the same result and working in time* $O(t + d)$*, where* d *is the depth of both P systems.* □

By using a variant of the proof techniques of Section 3.1 and 3.2 it is possible to employ even other accepting conditions. For instance, it is possible to keep the *last* output object (before halting) as the result of the computation, rather than the first one, by storing the last of the sequence of output objects in the charge of the outermost membrane, but only outputting it when the original P system halts. Even more generally, we can collect the sequence of output objects and combine them by applying any computable function (exploiting the universality of P systems).

4. Charges and priority

We will now show how a confluent P system Π with priority and any number of charges can be efficiently simulated by a confluent P system Π' without priority and using only two charges. The idea is to give a total ordering of the set of rules of Π compatible with its original priority, say $r_1 \succ r_2 \succ \cdots \succ r_m$; we decompose each computation step of Π into a sequence of m micro-steps, each one applying exactly one rule in the whole system as much as possible. The computation is thus sequential across the set of rules, but each rule r_i is applied in a maximally parallel way in all membranes involved in r_i. Halting in Π' is triggered by the halting of Π, assuming that each membrane of the latter system has charge $\spadesuit$, as proved possible in Section 3.2.

Notice that a linear priority does not make the P system Π deterministic, since send-in rules choose an arbitrary membrane among a set of different but externally indistinguishable ones having the same label (this will be the only form of nondeterminism for Π with priority $\succ$ and thus for Π'). On the other hand, using a total priority ordering of the rules requires, in general, the simulated P system Π to be confluent, since only a subset of its computations are simulated by Π'. Non-confluent P systems Π can be simulated using our construction if they already have a total priority ordering and, in that case, the simulating P system Π' is also non-confluent.

The membrane structure of Π' is, once again, identical to that of Π. A configuration $\mathcal{C}$ at time t of Π is encoded as a configuration $\mathcal{C}'$ at time $(6r + 6)t$ of Π' (i.e., every step of Π is simulated by $6r + 6$ steps of Π') as follows: if $\mathcal{C}$

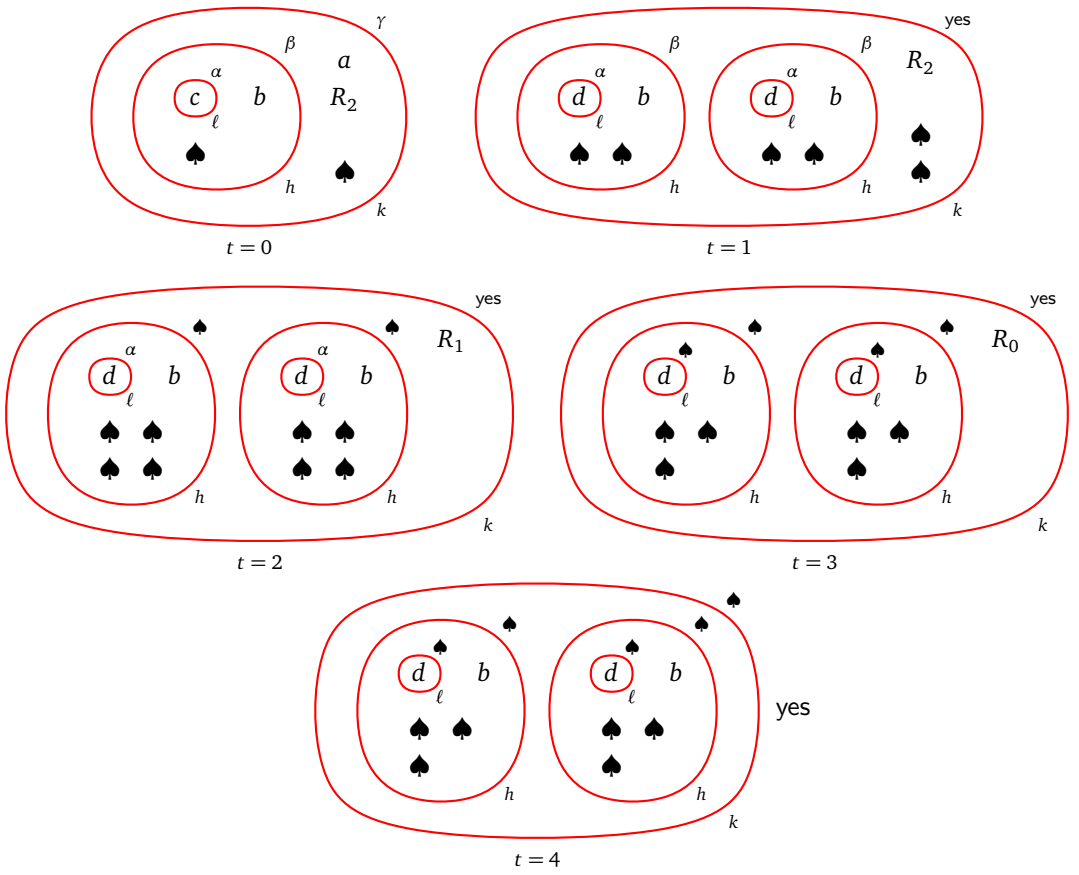

Figure 2: Simulation of a generalised P system Π with initial configuration $[[[c]^{\alpha}_{\ell}\ b]^{\beta}_{h}\ a]^{\gamma}_{k}$ and rules $[a]^{\gamma}_{k} \to [\ ]^{\delta}_{k}$ yes, $[b]^{\beta}_{h} \to [b]^{\beta}_{h}\ [b]^{\beta}_{h}$, and $[c \to d]^{\alpha}_{\ell}$. This P systems accepts by sending out yes without halting. The initial configuration of the simulating P system is shown at time $t = 0$. The object a is sent out without immediately producing the result object yes, but rather storing that information in the charge of k. At the same time, the other rules of the original P system are applied faithfully by the simulation, and the objects ♠ are doubled. Notice that the division rule $[b]^{\beta}_{h} \to [b]^{\beta}_{h}\ [b]^{\beta}_{h}$ would be applied forever, but the change of charge of the outermost membrane disables rule (6), enabling the lower priority rule (8) to send in an object ♠ inside each child membrane h and change their charge to ♠ ($t = 2$). This stops the application of the original rules, starts the recursive halting procedure for the internal membranes, and also triggers the countdown by object $R_d = R_2$. After $d = 2$ steps ($t = 3$) all membranes have stopped, except for the outermost one, that finally halts by sending out the timer object R_0 as yes.

contains a membrane having configuration $[w]^{\alpha}_{h}$, then the corresponding membrane in $\mathcal{C}'$ has configuration $[w\ \alpha]^{0}_{h}$, that is, the original charge is encoded as an object in $\mathcal{C}'$. We can view this as an invariant maintained by the simulation for all time steps t of Π. Notice that it is trivial to recover the original configuration $\mathcal{C}$ from $\mathcal{C}'$ and vice versa.

Simulating each step of Π begins with an initialisation phase of four steps of Π′. First we rewrite the charge-object α as $\alpha'\ \oplus$:

$$[\alpha \to \alpha'\ \oplus]^{0}_{h} \qquad \text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$

Each membrane of Π′ has now the configuration $[w\ \alpha'\ \oplus]^{0}_{h}$. The object $\oplus$ changes the charge of the membrane to + (it will always have this behaviour in the rest of the paper), while α' is rewritten into $\alpha''\ \odot$:

$$[\oplus]^{0}_{h} \to [\ ]^{+}_{h}\ \#$$
$$\text{for } h \in \Lambda$$
$$[\alpha' \to \alpha''\ \odot]^{0}_{h}$$
$$\text{for } h \in \Lambda \text{ and } \alpha \in \Psi$$

This leads to the membrane configuration $[w\ \alpha''\ \odot]^{+}_{h}$. The object $\odot$ changes the charge to 0 (here and in the

rest of the paper), while the objects in the original alphabet Γ gain a prime; the object α'' is rewritten into $\alpha''' \bullet$, where $\bullet$ is $\odot$ if rule r_1 has membrane h and charge α on the left-hand side, and $\bullet$ is $\oplus$ otherwise:

$$\begin{aligned} &[\odot]_h^+ \to [\ ]_h^0 \# && \text{for } h \in \Lambda \\ &[a \to a']_h^+ && \text{for } h \in \Lambda \text{ and } a \in \Gamma \\ &[\alpha'' \to \alpha''' \bullet]_h^+ && \text{for } h \in \Lambda \text{ and } \alpha \in \Psi \end{aligned}$$

The current membrane configuration is thus $[w'\ \alpha''' \bullet]_h^0$, where w' is w with all objects primed. The object $\bullet$ is then sent out, setting the charge of h to $+$ (if $\bullet$ is $\oplus$) or 0 (if $\bullet$ is $\odot$), while all remaining objects take a subscript 1:

$$\begin{aligned} &[\odot]_h^0 \to [\ ]_h^0 \# && \text{for } h \in \Lambda \\ &[\alpha''' \to \alpha_1]_h^0 && \text{for } h \in \Lambda \text{ and } \alpha \in \Psi \\ &[a' \to a_1]_h^0 && \text{for } h \in \Lambda \text{ and } a \in \Gamma \end{aligned}$$

This leads either to membrane configuration $[w_1\ \alpha_1]_h^0$ or $[w_1\ \alpha_1]_h^+$, depending on whether rule r_1 has the correct label and charge on the left-hand side.

We now establish a second invariant: for each $1 \le i \le m$, each membrane has one of four possible forms of configurations at time $(6r+6)(t-1)+6(i-1)+4$:

1. $[w_i\ \alpha_i]_h^0$ denotes that we have already tried to apply rules $r_1, \ldots, r_{i-1}$ in sequence (each of them in a maximally parallel way), and no blocking rule for h has been applied during the simulation of the current step of Π. The membrane contains the multiset of objects w_i, where each object has subscript i, and the charge of the simulated membrane is α. Furthermore, rule r_i has label h and charge α on the left-hand side.
2. $[w_i\ \alpha_i]_h^+$ is as in 1, but rule r_i has either the wrong label or the wrong charge on the left-hand side.
3. $[w_i\ \alpha_{i,j}]_h^0$ is as in 1, but a blocking rule r_j for h, for some $j < i$, has been previously applied during the simulation of the current step of Π, and r_i is necessarily an object evolution rule.
4. $[w_i\ \alpha_{i,j}]_h^+$ is as in 1, but a blocking rule r_j for h, for some $j < i$, has been previously applied during the simulation of the current step of Π, rule r_i is either an object evolution rule with wrong label or charge on the left-hand side, or it is a blocking rule.

Let us consider the four types of possible membrane configurations separately.

4.1. Membrane configuration of the form $[w_i\ \alpha_i]_h^0$

The behaviour of the P system Π' when a membrane configuration at time $(6r+6)(t-1)+6(i-1)+4$ is $[w_i\ \alpha_i]_h^0$ depends on the type of rule r_i of Π to be simulated.

4.1.1. Applicable send-out rules

Suppose that r_i is a send-out rule of Π of the form $[a]_h^\alpha \to [\ ]_h^\beta\ b$; also suppose that there is a membrane h that contains at least one instance of a, i.e., that the configuration of the membrane in Π' is $[a_i\ v_i\ \alpha_i]_h^0$ for some multiset v_i. The rule is implemented by first sending out an object a_i as $\widetilde{b}_i'$; the tilde here indicates an instance of object b_i that has already been subject to a rule during this simulated step of Π. The charge of the membrane is also changed to $+$ in order to signal that rule r_i was actually applied (i.e., that the membrane contained at least one instance of a_i):

$$[a_i]_h^0 \to [\ ]_h^+\ \widetilde{b}_i' \tag{9}$$

At the same time, the object α_i is primed:

$$[\alpha_i \to \alpha_i']_h^0$$

The configuration of the membrane is now $[v_i\ \alpha'_i]^+_h$, and the object $\widetilde{b}'_i$ is now managed by the outer membrane. When the membrane becomes positive, each subscripted original object of Γ (possibly in a tilded version) gains a prime, while α'_i becomes α''_i:

$$[c_i \to c'_i]^+_h \qquad \text{for } c \in \Gamma \tag{10}$$
$$[\widetilde{c}_i \to \widetilde{c}'_i]^+_h \qquad \text{for } c \in \Gamma \tag{11}$$
$$[\alpha'_i \to \alpha''_i]^+_h \tag{12}$$

This leads us to membrane configuration $[v'_i\ \alpha''_i]^+_h$, where v'_i is v_i with all objects primed. The object α''_i is now rewritten as follows:

$$[\alpha''_i \to \alpha'''_{i,i}]^+_h \tag{13}$$

thus storing in its second subscript the index i of the blocking rule that has actually been applied. The membrane configuration thus becomes $[v'_i\ \alpha'''_{i,i}]^+_h$. The object representing the charge α is rewritten again as:

$$[\alpha'''_{i,i} \to \alpha''''_{i,i}\ \odot]^+_h$$

Hence the configuration of the membrane is now $[v'_i\ \alpha''''_{i,i}\ \odot]^+_h$. The object $\odot$ sets the charge to 0:

$$[\odot]^+_h \to [\ ]^0_h\ \#$$

while $\alpha''''_{i,i}$ rewrites itself to $\alpha'''''_{i,i}$ and produces $\bullet$, where $\bullet$ is $\odot$ if rule r_{i+1} is an evolution rule with label h and charge α (i.e., r_{i+1} is potentially applicable), and $\bullet$ is $\oplus$ otherwise (i.e., r_{i+1} is not applicable due to the label, the charge, or the membrane h having already been used):

$$[\alpha''''_{i,i} \to \alpha'''''_{i,i}\ \bullet]^0_h$$

The membrane configuration is thus $[v'_i\ \alpha'''''_{i,i}\ \bullet]^0_h$. In the last step, we need to increase the rule counter i to $i+1$, remove all primes, and update the charge of h according to $\bullet$:

$$[\alpha'''''_{i,i} \to \alpha_{i+1,i}]^0_h$$
$$[c'_i \to c_{i+1}]^0_h \qquad \text{for } c \in \Gamma$$
$$[\widetilde{c}'_i \to \widetilde{c}_{i+1}]^0_h \qquad \text{for } c \in \Gamma$$
$$[\odot]^0_h \to [\ ]^0_h\ \#$$
$$[\oplus]^0_h \to [\ ]^+_h\ \#$$

We have thus reached membrane configuration $[v_{i+1}\ \alpha_{i+1,i}]^0_h$ or $[v_{i+1}\ \alpha_{i+1,i}]^+_h$ after 6 steps of Π', thus restoring the invariant.

4.1.2. Applicable send-in rules

If r_i is a send-in rule $a\ [\ ]^\alpha_h \to [b]^\beta_h$ and the outer membrane contains an instance of object a that is actually assigned to this rule for the current membrane, the computation proceeds similarly to the one for applicable send-out rules, in 6 steps, except that rule (9) is replaced by the rule

$$a'_i\ [\ ]^0_h \to [\widetilde{b}_i]^+_h$$

which is applied one step later, since the object a_i must first be primed. This happens when the charge α is represented by α'_i, which applies rule (12) and, one step later, records the application of the send-in rule by applying (13).

4.1.3. Applicable division rules

If r_i is a weak (elementary or non-elementary) division rule $[a]_h^\alpha \to [b]_h^\beta\ [c]_h^\gamma$ and membrane h contains an instance of a, then the computation evolves again as for applicable send-out rules, in 6 steps, with the following variations. Rule (9) is replaced by the send-out rule

$$[a_i]_h^0 \to [\ ]_h^+ \#$$

This rule sets the charge to +, thus signalling that the rule is actually being applied. This fact is recorded by the object α_i'' using rule (13); the actual division does not happen immediately, but is delayed until the end of the iteration across all rules. The reason for this delay is to comply with the usual semantics of P systems, where internal membranes logically evolve before external ones: if division happened immediately, the internal membranes may evolve differently in the two copies of the membrane, due to the nondeterminism possibly introduced by send-in rules.

4.1.4. Applicable dissolution rules

A dissolution rule $r_i = [a]_h^\alpha \to b$ is simulated in 6 steps as a send-out rule followed by a delayed dissolution; this is recorded, as for division rules, in the second subscript of the object $\alpha_{i,i}'''$. The actual dissolution is delayed because further object evolution rules might be applicable.

4.1.5. Object evolution rules

An object evolution rule $r_i = [a \to x]_h^\alpha$, with $x \in \Gamma^\star$, is simulated in a slightly different way than blocking rules: since it is applied in parallel to all objects a contained in h, this rule cannot change the charge of the membrane to signal its application. The object α_i must thus evolve without knowing the number of objects to which rule r_i is applied, if any.

In the first step the actual evolution occurs:

$$\begin{aligned} &[a_i \to \widetilde{x}_i]_h^0 \\ &[\alpha_i \to \alpha_i']_h^0 \end{aligned} \tag{14}$$

where $\widetilde{x}_i$ is x with all objects tilded and subscripted by i. This leads to membrane configuration $[v_i\ \alpha_i']_h^0$, where v_i is the multiset w_i updated according to rule (14). In the second and third steps the following rules are applied one after the other:

$$\begin{aligned} &[\alpha_i' \to \alpha_i'']_h^0 \\ &[\alpha_i'' \to \alpha_i''' \oplus]_h^0 \end{aligned}$$

leading to membrane configuration $[v_i\ \alpha_i''' \oplus]_h^0$. The charge is then set to +, while object α_i''' further evolves into α_i'''' and $\odot$:

$$\begin{aligned} &[\oplus]_h^0 \to [\ ]_h^+ \# \\ &[\alpha_i''' \to \alpha_i'''' \odot]_h^0 \end{aligned}$$

leading to configuration $[v_i\ \alpha_i'''' \odot]_h^+$. The objects in Γ and their tilded versions are now primed, while the charge becomes 0 and object α_i'''' gains another prime:

$$\begin{aligned} &[c_i \to c_i']_h^+ && \text{for } c \in \Gamma \\ &[\widetilde{c}_i \to \widetilde{c}_i']_h^+ && \text{for } c \in \Gamma \\ &[\odot]_h^+ \to [\ ]_h^0 \# \\ &[\alpha_i'''' \to \alpha_i''''' \bullet]_h^+ \end{aligned}$$

where $\bullet$ works as described above. This leads to membrane configuration $[v_i'\ \alpha_i''''' \bullet]_h^0$. The last step is as for applicable send-out rules, and leads to $[v_{i+1}\ \alpha_{i+1}]_h^0$ or $[v_{i+1}\ \alpha_{i+1}]_h^+$ depending on r_{i+1}.

4.1.6. Non-applicable blocking rules

If r_i is a blocking rule with label h and charge α, but the object on the left-hand side of the rule is missing, the membrane reaches configuration $[v_i\ \alpha_i']_h^0$ after one step, where v_i is w_i except for any objects coming from or sent in children membranes. The object α_i'' detects that rule r_i was not applied by observing the neutral charge of the membrane. The membrane can then evolve as for object evolution rules, leading to membrane configuration $[v_{i+1}\ \alpha_{i+1}]_h^0$ or $[v_{i+1}\ \alpha_{i+1}]_h^+$ (depending on r_{i+1}) in 6 computation steps.

4.2. Membrane configuration of the form $[w_i\ \alpha_i]_h^+$

If the membrane reaches configuration $[w_i\ \alpha_i]_h^+$, then rule r_i is not applicable either because it involves a label different from h, or a charge different from α on the left-hand side. In that case, the membrane must reach configuration $[v_{i+1}\ \alpha_{i+1}]_h^0$ (or $[v_{i+1}\ \alpha_{i+1}]_h^+$ if r_{i+1} has the wrong label or charge), where v is w except for any object coming from or sent in children membranes, after exactly 6 steps, in order to keep all membranes synchronised. In this membrane configuration, as in the previous cases, the objects of Γ and their tilded counterparts, possibly coming from a children membrane, have their subscript incremented after being primed by rules (10) and (11).

First the object α_i waits for three steps, and then produces a $\odot$; the object α_i is also tilded to record the fact that rule r_i is not being applied at this time:

$$[\alpha_i \to \widetilde{\alpha}_i']_h^+ \qquad [\widetilde{\alpha}_i' \to \widetilde{\alpha}_i'']_h^+ \qquad [\widetilde{\alpha}_i'' \to \widetilde{\alpha}_i''']_h^+ \qquad [\widetilde{\alpha}_i''' \to \widetilde{\alpha}_i'''' \odot]_h^+$$

While the charge is set to 0, the object $\bullet$ (which is either $\odot$ or $\oplus$, according to the label and charge of r_{i+1}) is produced:

$$[\odot]_h^+ \to [\ ]_h^0\ \# \qquad\qquad [\widetilde{\alpha}_i'''' \to \widetilde{\alpha}_i''''\ \bullet]_h^+$$

Finally, all subscripts are incremented:

$$\begin{aligned} &[\widetilde{\alpha}_i'''' \to \alpha_{i+1}]_h^0 \\ &[c_i' \to c_{i+1}]_h^0 && \text{for } c \in \Gamma \\ &[\widetilde{c}_i' \to \widetilde{c}_{i+1}]_h^0 && \text{for } c \in \Gamma \end{aligned}$$

This leads to the membrane configuration for rule r_{i+1}.

4.3. Membrane configuration of the form $[w_i\ \alpha_{i,j}]_h^0$

If the membrane has a configuration of the form $[w_i\ \alpha_{i,j}]_h^0$, then a blocking rule r_j, with $j < i$, has already been applied to that membrane, and r_i is thus necessarily an object evolution rule with label h and charge α. The computation then proceeds as for object evolution rules in Section 4.1, except that the second subscript j of $\alpha_{i,j}$ is also preserved, thus reaching membrane configuration $[v_{i+1}\ \alpha_{i+1,j}]_h^0$ (or $[v_{i+1}\ \alpha_{i+1,j}]_h^+$) after 6 steps, where v is w updated according to r_i.

4.4. Membrane configuration of the form $[w_i\ \alpha_{i,j}]_h^+$

If the membrane has a configuration of the form $[w_i\ \alpha_{i,j}]_h^+$, then a blocking rule r_j, with $j < i$, has already been applied to that membrane, and r_i either has the wrong label or charge, or it is another blocking rule (and thus it is not applicable). In this case, the computation proceeds as in Section 4.2, except that the second subscript j of $\alpha_{i,j}$ is preserved.

Example 5. Figure 3 shows how the rules $r_i = d\ [\ ]_h^\alpha \to [f\ ]_h^\delta$ and $r_{i+1} = [b]_h^\alpha \to [\ ]_h^\gamma\ b$ are simulated (respectively from $t = 0$ to $t = 6$ and $t = 6$ to $t = 12$) starting from the configuration $\left[[b]_h^\alpha\ [b\ c]_h^\alpha\ [b]_h^\beta\ e\ d\right]_k^\gamma$. Notice how the upper left membrane h applies r_i, storing that information in the second subscript of its charge-object (this also allows the charge-object to be updated as δ, which happens only after having simulated all rules for this computation step). That membrane is now unavailable for the application of rule r_{i+1} (i.e., its charge is + at time $t = 6$, rather than 0 as before applying r_i). The lower left membrane h has charge + both at $t = 0$ and at $t = 6$, since both rules r_i and r_{i+1} require the charge α rather than β. Starting from the configuration at time $t = 12$, the rule r_{i+2} is simulated; by observing the charges of the membranes, it may be deduced that this rule is applicable in the outermost membrane, since its charge is 0.

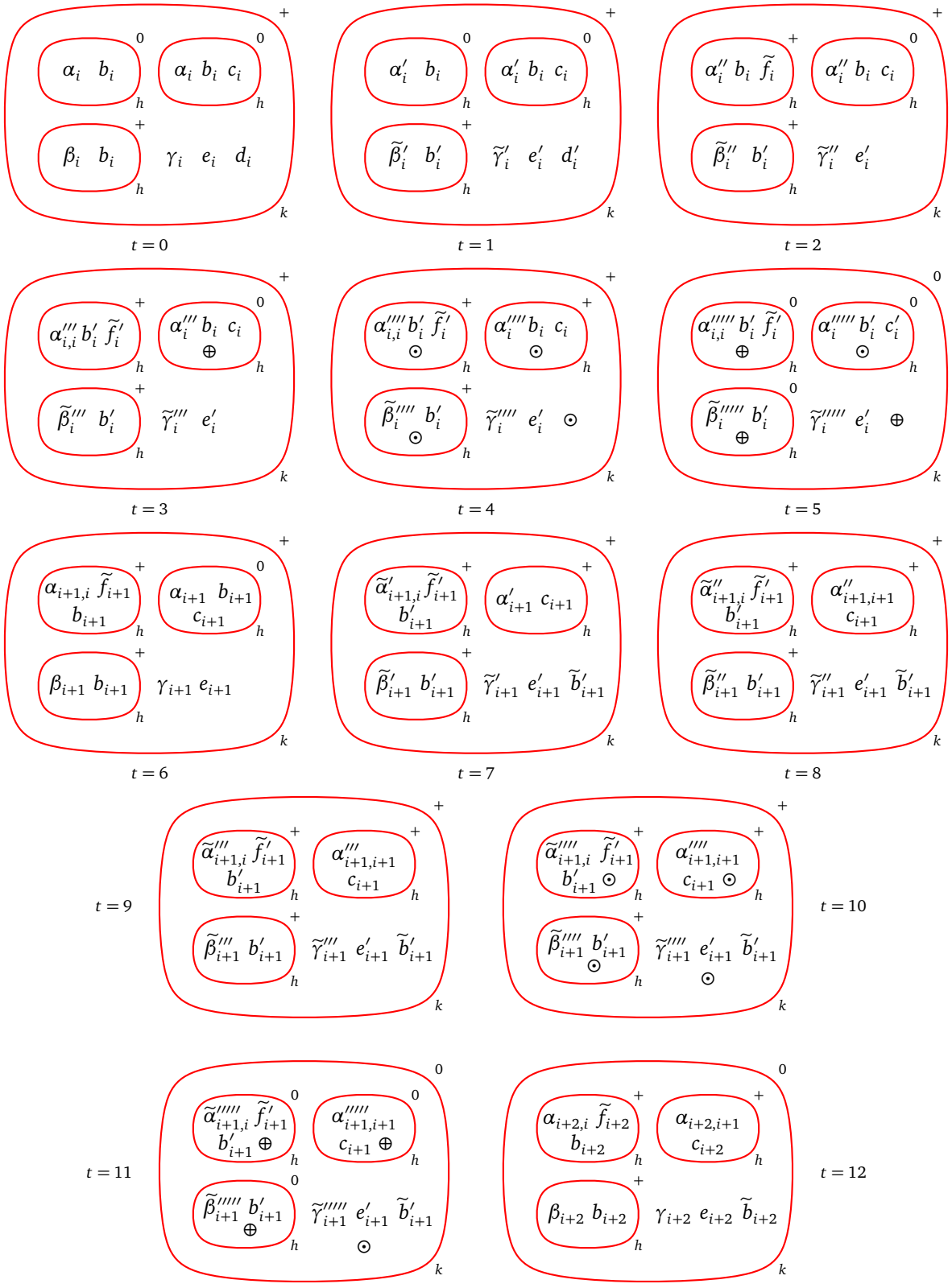

Figure 3: Simulation of a P system with generalised charges and rule priority by a standard P system with active membranes.

4.5. Concluding the simulation of one step

After having simulated all rules $r_1 \succ r_1 \succ \cdots \succ r_m$ in priority order, the subscripts of the objects inside each membrane will reach the value $m+1$. Suppose that all membranes have been set to neutral at that time (as if the non-existing rule r_{m+1} was always applicable). In order to restore the outer invariant of Section 4, we need to remove the subscripts and the tildes, update the objects representing the charges, and complete the application of dissolution and division rules.

The objects in Γ and their tilded counterparts can be immediately rewritten into their final form:

$$[c_{m+1} \to c]_h^0 \qquad \text{for } h \in \Lambda \text{ and } c \in \Gamma$$
$$[\widetilde{c}_{m+1} \to c]_h^0 \qquad \text{for } h \in \Lambda \text{ and } c \in \Gamma$$

If any dissolution rule r_j was applied by the simulated P system during this simulated step, the actual dissolution can now take place (recall that the object b on the right-hand side of the rule has already been sent out):

$$[\alpha_{m+1,j}]_h^0 \to \#$$

Analogously, if a weak (elementary or non-elementary) division rule r_j involving membrane h was applied, then we can first perform the actual division:

$$[\alpha_{m+1,j}]_h^0 \to [\alpha'_{m+1,j}]_h^0\, [\alpha''_{m+1,j}]_h^0$$

and then update the simulated charges and create the right-hand side objects in the two copies of h:

$$[\alpha'_{m+1,j} \to \beta\ b]_h^0 \qquad\qquad [\alpha''_{m+1,j} \to \gamma\ c]_h^0$$

If a blocking rule r_j of the remaining types (send-in or send-out) was applied to the membrane, then we just need to update the charge object to β, the right-hand side charge of r_j; however, this must take two steps to maintain synchronisation with membranes where division was applied:

$$[\alpha_{m+1,j} \to \alpha'_{m+1,j}]_h^0 \qquad\qquad [\alpha'_{m+1,j} \to \beta]_h^0$$

Finally, if no blocking rule was applied in membrane h, then we must keep the same charge as in the previous step of Π; once again, this must take two steps to maintain all membranes synchronised:

$$[\alpha_{m+1} \to \alpha'_{m+1}]_h^0 \qquad [\alpha'_{m+1} \to \alpha]_h^0 \qquad \text{for } h \in \Lambda,\ \alpha \in \Psi$$

The new configuration of Π' then corresponds to a reachable configuration of Π according to the encoding described above.

4.6. Halting and output

In the simulation of this section, detecting whether a membrane of Π has stopped computing paradoxically requires us to iterate across all rules, thus preventing the simulating P system Π' to halt. However, according to the results of Section 3, we can always assume the simulated P system Π to be a standard recogniser, and that all membranes have charge ♠ when they stop computing. Thus, we simulate each membrane as described above until it assumes the charge ♠. When this happens, the simulating membrane contains the object ♠; however, by construction the children of this membrane, if any, have not yet assumed the charge ♠, as this will propagate there by send-in in the next computation step. Hence, when a simulated membrane gets the charge ♠, we must perform a last iteration across the rules of Π in order to simulate those send-in rules, and then the simulating membrane can finally halt. This last iteration is needed in order to update the subscripts of the objects c_i (with $c \in \Gamma$). Another small modification to be made involves sending out the yes or no object: in Π, the corresponding rules do not generally have the lowest priority, and thus the actual sending out of the result object must be delayed in order to be the last action performed by the simulating P system Π'.

Halting the simulation of a membrane can thus be performed by simply deleting the object $\spadesuit_{m+1}$ obtained after the last iteration across the rules, that is, the object $\spadesuit$ of construction of Section 3.2 with the subscript $m+1$ added in Section 4.5, denoting the fact that the m rules of Π have been simulated:

$$[\spadesuit_{m+1} \to \epsilon]^0_h \qquad \text{for } h \in \Lambda \tag{15}$$

The outputting of yes or no by Π' at the last step can be achieved by replacing any outermost membrane output rules r_j of the forms

$$[a]^\alpha_k \to [\ ]^\beta_k \text{ yes} \qquad\qquad [a]^\alpha_k \to [\ ]^\beta_k \text{ no}$$

of Π by a rule of the form

$$[a]^\alpha_k \to [\ ]^\beta_k \ \# \qquad\qquad [a]^\alpha_k \to [\ ]^\beta_k \ \#$$

During the subscript-deleting phase of Section 4.5 we can perform the actual output by using one of the following rules:

$$[\alpha_{m+1,j}]^0_k \to [\ ]^0_k \text{ yes} \qquad [\alpha_{m+1,j}]^0_k \to [\ ]^0_k \text{ no} \qquad \text{for } 1 \le j \le m \tag{16}$$

Since, by hypothesis, the rest of the P system Π has already halted, the simulation of Π' in the worst case completes the last iteration across the rules of Π for the innermost membranes by applying a rule of type (15) exactly when an output rule (16) is applied. The P system Π' halts immediately when these rules complete.

4.7. Main result

The only remaining detail to consider is the amount of resources needed in order to perform the simulations described in this section and in Section 3. It suffices to observe that all rules of the final P system are obtained by repeating simple patterns with parameters ranging over sets of polynomial size with respect to the description of the simulated P system (e.g., the set of rules of the original P system, its set of labels, the set of integers up to the membrane nesting depth, ...).

For example, the rules of type (8) can be output by using two nested loops as follows:

for $h \in \Lambda$ **do**
  for $\alpha \in \Psi$ **do**
    output "$\spadesuit\ [\ ]^\alpha_h \to [\#]^\spadesuit_h$"
  end for
end for

The construction of Π' can thus be performed in polynomial time. This leads immediately to our main result:

Theorem 6. *Let Π be a generalised confluent recogniser P system using priority and generalised charges, working in time t. Then, there exists a standard confluent recogniser P system Π' without priority and using only two charges having the same result as Π and working in time $O(r \times (d + t))$, where r is the number of rules of Π and d is the depth of its membrane structure. Furthermore, the mapping $\Pi \mapsto \Pi'$ can be computed in polynomial time with respect to the length of the description of Π.* □

4.8. A note on rule types

The construction used to prove Theorem 6 necessarily requires evolution, send-in and send-out rules. Any other type of rule (dissolution, elementary and weak non-elementary division) is only necessary if the original P system Π being simulated employs it. This construction might, in principle, be extended in order to simulate other kinds of rules, provided that the simulating P system Π' is also allowed to use them. The technical details are, however, necessarily dependent on the specific type of rule.

5. Conclusions

The results of this paper have shown that the number of charges (as long as it is at least two) and the exact accepting conditions of recogniser P systems with active membranes are immaterial, and can always be reduced to two charges and to the standard definition of recogniser P system without loss of efficiency. This allows us to use as many charges as are convenient for the solution of the current problem, to employ more relaxed halting conditions, and even to add a rule priority. Hopefully, these tools will yield algorithms having less auxiliary technical details and a better focus on the novel techniques and ideas employed.

We conjecture that results analogous to those presented in this paper may also be proved for other classes of P systems, therefore further simplifying membrane computing algorithms. For instance, it would be interesting to explore which features (such as charges, rule priorities and accepting conditions) may be added to tissue P systems [19] without changing their computing power or efficiency.

Another interesting research problem is the existence of results analogous to those presented here for P systems using rule application strategies different from the classic maximal parallelism. Since the techniques employed here are based on iterating synchronously over the set of rules in all membranes of the P system, it seems unlikely that they could be extended to variants such as purely asynchronous [8], minimally parallel P systems (where at least one rule per region is applied, without further constraints [7]) or time-free P systems (which must provide the correct result independent of the duration of each rule [5]). Hence, alternative proof techniques would have to be devised for these classes of P systems.

Acknowledgements

This work was partially supported by Fondo d'Ateneo (FA) 2015 of Università degli Studi di Milano-Bicocca: "Complessità computazionale e applicazioni crittografiche di modelli di calcolo bioispirati".

References

[1] A. Alhazov, R. Freund, On the efficiency of P systems with active membranes and two polarizations, in: G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, 5th International Workshop, WMC 2004, volume 3365 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 146–160. URL: http://dx.doi.org/10.1007/978-3-540-31837-8_8.

[2] A. Alhazov, R. Freund, A. Riscos-Núñez, One and two polarizations, membrane creation and objects complexity in P systems, in: D. Zaharie, D. Petcu, V. Negru, T. Jebelean, G. Ciobanu, A. Cicortas, A. Abraham, M. Paprzycki (Eds.), Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC'05, IEEE, 2005, pp. 385–394. URL: http://dx.doi.org/10.1109/SYNASC.2005.54.

[3] A. Alhazov, A. Leporati, G. Mauri, A.E. Porreca, C. Zandron, Space complexity equivalence of P systems with active membranes and Turing machines, Theoretical Computer Science 529 (2014) 69–81. URL: http://dx.doi.org/10.1016/j.tcs.2013.11.015.

[4] A. Alhazov, M.J. Pérez-Jiménez, Uniform solution to QSAT using polarizationless active membranes, in: J. Durand-Lose, M. Margenstern (Eds.), Machines, Computations, and Universality, 5th International Conference, MCU 2007, volume 4664 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 122–133. URL: http://dx.doi.org/10.1007/978-3-540-74593-8_11.

[5] M. Cavaliere, D. Sburlan, Time-independent P systems, in: G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, 5th International Workshop, WMC 2004, volume 3365 of *Lecture Notes in Computer Science*, pp. 239–258. URL: http://dx.doi.org/10.1007/978-3-540-31837-8_14.

[6] G. Ciobanu, S. Marcus, Gh. Păun, New strategies of using the rules of a P system in a maximal way: Power and complexity, Romanian Journal of Information Science and Technology 12 (2009) 157–173. URL: http://www.imt.ro/romjist/Volum12/Number12_2/cuprins12_2.htm.

[7] G. Ciobanu, L. Pan, Gh. Păun, M.J. Pérez-Jiménez, P systems with minimal parallelism, Theoretical Computer Science 378 (2007) 117–130. doi:http://dx.doi.org/10.1016/j.tcs.2007.03.044.

[8] R. Freund, Asynchronous P systems and P systems working in the sequential mode, in: G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, 5th International Workshop, WMC 2004, volume 3356 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 36–62. URL: http://dx.doi.org/10.1007/978-3-540-31837-8_3.

[9] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Nuñez, F.J. Romero-Campero, Computational efficiency of dissolution rules in membrane systems, International Journal of Computer Mathematics 83 (2006) 593–611. URL: http://dx.doi.org/10.1080/00207160601065413.

[10] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Membrane division, oracles, and the counting hierarchy, Fundamenta Informaticae 138 (2015) 97–111. URL: http://dx.doi.org/10.3233/FI-2015-1201.

[11] N. Murphy, D. Woods, The computational power of membrane systems under tight uniformity conditions, Natural Computing 10 (2011) 613–632. URL: http://dx.doi.org/10.1007/s11047-010-9244-7.

[12] N. Murphy, D. Woods, Uniformity is weaker than semi-uniformity for some membrane systems, Foundamenta Informaticae 134 (2014) 129–152. URL: http://dx.doi.org/10.3233/FI-2014-1095.

[13] Gh. Păun, Computing with membranes, Journal of Computer and System Sciences 61 (2000) 108–143. URL: http://dx.doi.org/10.1006/jcss.1999.1693.

[14] Gh. Păun, P systems with active membranes: Attacking NP-complete problems, Journal of Automata, Languages and Combinatorics 6 (2001) 75–90.

[15] Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), The Oxford Handbook of Membrane Computing, Oxford University Press, 2010.

[16] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Complexity classes in models of cellular computing with membranes, Natural Computing 2 (2003) 265–284. URL: http://dx.doi.org/10.1023/A:1025449224520.

[17] A.E. Porreca, A. Leporati, G. Mauri, C. Zandron, P systems with elementary active membranes: Beyond NP and coNP, in: M. Gheorghe, T. Hinze, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, 11th International Conference, CMC 2010, volume 6501 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 338–347. URL: http://dx.doi.org/10.1007/978-3-642-18123-8_26.

[18] Gh. Păun, Introduction to membrane computing, in: G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (Eds.), Applications of Membrane Computing, Natural Computing Series, Springer, 2006, pp. 1–42. URL: http://dx.doi.org/10.1007/3-540-29937-8_1.

[19] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos Núñez, Tissue P systems with cell division, International Journal of Computers, Communications & Control 3 (2008) 295–303. URL: http://dx.doi.org/10.15837/ijccc.2008.3.2397.

[20] P. Sosík, The computational power of cell division in P systems: Beating down parallel computers?, Natural Computing 2 (2003) 287–298. URL: http://dx.doi.org/10.1023/A:1025401325428.

[21] C. Zandron, C. Ferretti, G. Mauri, Solving NP-complete problems using P systems with active membranes, in: I. Antoniou, C.S. Calude, M.J. Dinneen (Eds.), Unconventional Models of Computation, UMC'2K, Proceedings of the Second International Conference, Springer, 2001, pp. 289–301. URL: http://dx.doi.org/10.1007/978-1-4471-0313-4_21.

[22] C. Zandron, A. Leporati, C. Ferretti, G. Mauri, M.J. Pérez-Jiménez, On the computational efficiency of polarizationless recognizer P systems with strong division and dissolution, Fundamenta Informaticae 87 (2008) 79–91. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi87-1-06.