A Turing Machine Simulation by P Systems without Charges

Alberto Leporati[1], Luca Manzoni[1,2], Giancarlo Mauri[1], Antonio E. Porreca[1,3], and Claudio Zandron[1]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336, 20126 Milano, Italy
{alberto.leporati,giancarlo.mauri,claudio.zandron}@unimib.it

[2] Dipartimento di Matematica e Geoscienze
Università degli Studi di Trieste
Via Valerio 12/1, 34127 Trieste, Italy
lmanzoni@units.it

[3] Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
antonio.porreca@lis-lab.fr

Summary. It is known that the kind of polarizationless P systems involved in the definition of the P conjecture are able to solve problems in the complexity class P by leveraging their uniformity condition. Here we show that they are indeed able to simulate a deterministic Turing machine working in polynomial time with a weaker uniformity condition and using only one level of membrane nesting. This allow us to embed this construction into more complex membrane structures, possibly showing that constructions similar to the one performed for P systems with charges can be carried on.

1 Introduction

P systems with active membranes are one of the model in the vast and diverse family of P systems, initially founded by Gh. Păun [6]. In this kind of P systems the space is delimited in different (possibly nested) regions via membranes, mimicking the way cellular membranes separate the inner part of a cell from the external environment. The biological inspiration does not end here: inside each membrane multiple objects, representing chemical substances, are transformed by rewriting rules, mimicking biochemical reactions. Moreover, each membrane may have an associated electrical charge, which typically can be negative, positive, or neutral, that affects the applicability of the rules embedded in the membrane. The communication between different regions of space is ensured by the ability of substances to move in and out of membranes (also depending on the membrane charge) and, possibly, to even dissolve a membrane, making all its content "fall out" in the containing region. The name "P systems with active membranes" stems from the fact that membranes play an active role during the computations, either by influencing the rules to be applied through charges, or by modifying the membrane hierarchy through membrane division or dissolution.

During the two decades following their introduction, P systems with active membranes have been employed to solve classically intractable problems, like NP-complete ones [10] or, more recently, problems in the complexity class $\mathsf{P}^{\#\mathsf{P}}$ and in the entire counting hierarchy [1]. To reach these results, the simulation of Turing machines (TM) provide an important building block. In particular, the construction of P systems simulating TM using as few membranes (or cells) as possible and limiting the depth of the system is one of the "tricks" that allows the nesting of multiple machines in order to solve problems in large complexity classes. For example, nesting of non-deterministic machines (where the non-determinism was simulated by membrane division) and a counting mechanism allow to characterize $\mathsf{P}^{\#\mathsf{P}}$, the class of all problems solvable by a deterministic TM with access to a $\#\mathsf{P}$ oracle [1, 3]. The same ideas can be applied to tissue P systems [4], where the different communication topology makes even more important to keep TM simulations compact [2].

The P conjecture is a long-standing open problem in membrane computing, first presented in 2005 [8, Problem F] that, in its essence, asks what is the power of one charge when compared to two charges. We feel that one important step to determine the computational power of active membrane systems without charges and with membrane dissolution is to see which is the minimal system able to simulate a deterministic polynomial-time TM. Here we show that a shallow system is sufficient to perform such a simulation by delegating only a minimal amount of work to the Turing machines involved in the uniformity condition. Hopefully, this construction will allow us to define systems in which different TM can be "embedded" at different levels in a large membrane structure, thus making possible to mimic the existing constructions performed for P systems with charges.

One specific application of this construction is the replication of the result presented in [1] concerning P systems with charges to the case of P systems without charges. In that paper we presented a construction of "nested oracles" each of them being, essentially, a simulation of a non-deterministic TM with some additional "plumbing" to perform some kind of interaction between the different membranes. The ability to perform this TM simulation without charges is an important step in porting the same construction to an apparently weaker model of P systems.

This paper is organized as follows: Section 2 will recall some basic notions on P systems. The main construction and result is presented in Section 3, while ideas for further research are presented in Section 4.

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* [9]. Here we just recall the formal definition of P systems with active membranes, without charges [7, 11].

Definition 1. *A polarizationless* P system with active membranes with dissolution 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;*
- *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 each of the d regions of μ;*
- *R is a finite set of rules.*

The rules in R are of the following types:

(a) *Object evolution rules*, of the form $[a \to w]_h$.
They can be applied inside a membrane labelled by h 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 $a\ [\ ]_h \to [b]_h$.
They can be applied to a membrane labelled by h and such that the parent region, i.e., the one containing membrane h, contains an occurrence of the object a; the object a is sent into h, becoming b.
(c) *Send-out communication rules*, of the form $[a]_h \to [\ ]_h\ b$.
They can be applied to a membrane labelled by h and containing an occurrence of the object a; the object a is sent out from h to the parent region, becoming b.
(d) *Dissolution rules*, of the form $[a]_h \to b$.
They can be applied to any membrane except the outermost one labelled by h and containing an occurrence of the object a; the object a is sent out from h to the parent region becoming b, the membrane h ceases to exist and all the other objects it contains are sent into the parent region.

A computation step changes the current configuration of the system 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 or communication rules must be subject to exactly one of them. Analogously, each membrane can only be subject to one communication or dissolution rule (types (b)–(d)) per computation step; for this reason, these rules will be called *blocking rules* in the rest of the paper. As a result, the only objects and membranes that do not evolve are those associated with no rule.
- 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 its internal configuration (including, recursively, the configurations of the membrane substructures it contains) has been updated.
- Any object sent out from it cannot re-enter the system again.

A *halting computation* of the P system Π is a finite sequence $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ 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 $\mathcal{C}_k$.

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*.

In order to solve decision problems (or, equivalently, decide languages), we use *families* of recogniser P systems $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$. 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 [5].

Definition 2. *A family of P systems $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ 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 a specific membrane, called the* input membrane.

Any explicit encoding of Π_x is allowed as output of the construction, as long as it is at most polynomially shorter than the one where the rules are listed one by one, the membrane structure is represented in such a way that all membranes are listed one by one and their content is encoded in unary. 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 [5] for further details on the encoding of P systems.

Among all possible uniformity conditions, obtained by imposing constrains on the Turing machines E and F, we are interested in (L, L)-uniform families of P systems, where both the machine constructing the P system given the size of the input in unary and the machine encoding the input can only employ logarithmic space. This uniformity condition is weaker than the usual (P, P)-uniformity, but it is needed to avoid that the computation which is intended to be performed by the P system is instead performed by the Turing machines E and F building it.

3 Simulation of Polynomial-time Turing Machines

The main idea of this section is to provide a simulation of a deterministic TM working in polynomial time by using a P system with only one level of nesting, i.e., by what is usually called a *shallow* P system.

A first observation is that two objects can meaningfully influence each other only through dissolution: while two objects might interfere due to the blocking nature of communication rules, any such interaction is actually not significant due to the confluence of the P system. Therefore, information between objects must be exchanged by performing dissolution, for example by having one of the objects "count" the steps needed before "falling out" in the parent region.

Let M be a polynomial-time deterministic TM having alphabet Σ, set of states Q, and transition function $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, +1\}$. We assume that, for an input of length n machine M halts in time $p(n)$ and, thus, it uses no more than $p(n) + 1$ cells. We are going to define a P system Π that simulates the computation of M in $O(p(n)|\Sigma|)$ steps. That is, the simulation of every step of M will require a number of steps in Π that is proportional to the size of the alphabet of M, thus providing an efficient simulation, i.e., a simulation that is only polynomially slower than the simulated system in terms of number of steps.

The P system Π has $(p(n) + 1)^2 + p(n)^2 + p(n) + 1$ labels, one for the skin membrane and two for each pair of time and position in the TM tape:

$$\begin{aligned} \Lambda = &\{0\} \cup \{(i, j) \mid i, j \in \{0, \ldots, p(n)\}\} \\ &\cup \{(i, j)' \mid i \in \{0, \ldots, p(n)\}\ j \in \{0, \ldots, p(n) - 1\}\} \quad . \end{aligned}$$

$$a_i\ [\ ]_{(i,0)} \to [a_{i,0,0}]_{(i,0)} \qquad \text{for } a \in \Sigma \tag{1}$$
$$[q^I \to q^I_{0,0,0}]_0 \tag{2}$$
$$[a_{i,j,k} \to a_{i,j,k+1}]_{(i.j)} \qquad \text{for } 0 \le k < \varphi(a) \text{ and } a \in \Sigma \tag{3}$$
$$[a_{i,j,k}]_{(i,j)} \to a_{i,j,k+1} \qquad \text{for } k = \varphi(a) \text{ and } a \in \Sigma \tag{4}$$
$$[a_{i,j,k} \to a_{i,j,k+1}]_0 \qquad \text{for } \varphi(a) < k \le m \text{ and } a \in \Sigma \tag{5}$$
$$q_{i,j,0}\ [\ ]_{(i,j)} \to [q_{i,j,1}]_{(i,j)} \qquad \text{for } q \in Q \tag{6}$$
$$[q_{i,j,k} \to q_{i,j,k+1}]_{(i,j)} \qquad \text{for } 1 \le k \le m \text{ and } q \in Q \tag{7}$$
$$[q_{i,j,k} \to q_{i,j,k+1,\varphi^{-1}(k)}]_0 \qquad \text{for } 1 \le k \le m, \text{ and } q \in Q \tag{8}$$
$$[q_{i,j,k,a} \to q_{i,j,k+1,a}]_0 \qquad \text{for } 1 \le k \le m,\ a \in \Sigma, \text{ and } q \in Q \tag{9}$$
$$a_{i,j,m+1}\ [\ ]_{(i,j)'} \to [a_{i,j,m+2}]_{(i,j)'} \qquad \text{for } a \in \Sigma \tag{10}$$
$$[q_{i,j,m+1,a} \to q_{i,j,m+2,a}]_0 \qquad \text{for } q \in Q \text{ and for } a \in \Sigma \tag{11}$$
$$[a_{i,j,m+2} \to a_{i,j,m+3}]_{(i,j)'} \qquad \text{for } a \in \Sigma \tag{12}$$
$$q_{i,j,m+2,a}\ [\ ]_{(i,j)'} \to [q_{i,j,m+3,a}]_{(i,j)'} \qquad \text{for } q \in Q \text{ and } a \in \Sigma \tag{13}$$
$$[a_{i,j,m+3} \to a_{i,j,m+4}]_{(i,j)'} \qquad \text{for } a \in \Sigma \tag{14}$$
$$[q_{i,j,m+3,a}]_{(i,j)'} \to q_{i,j,m+4,a} \qquad \text{for } q \in Q \text{ and } a \in \Sigma \tag{15}$$
$$[a_{i,j,m+4}]_{(i,j)'} \to a_{i,j,m+5} \qquad \text{for } a \in \Sigma \tag{16}$$
$$[a_{i,j,m+4} \to \epsilon]_0 \qquad \text{for } a \in \Sigma \tag{17}$$
$$[q_{i,j,m+4,a} \to q_{i,j,m+5,a}\ b_{i+d,j,m+5}]_0 \qquad \text{for } q \in Q,\ a \in \Sigma, \text{ and } \delta(q,a) = (r,b,d) \tag{18}$$
$$a_{i,j,m+5}\ [\ ]_{(i,j+1)} \to [a_{i,j+1,0}]_{(i,j+1)} \qquad \text{for } a \in \Sigma \tag{19}$$
$$[q_{i,j,m+5,a} \to r_{i+d,j+1,0}]_0 \qquad \text{for } q \in Q,\ a \in \Sigma, \text{ and } \delta(q,a) = (r,b,d) \tag{20}$$

Fig. 1. The complete set of rules employed by the P system Π that simulates the Turing machine M, here given as a handy reference. The numbering is the same as the one used in the text when introducing each rule.

Since we assume that no kind of membrane division is present, in the following we can identify membranes and labels, since each label is used by exactly one membrane. The semantics of the labels is that a membrane with label (i,j) will represent the i-th cell of the TM tape at time j. The additional membrane $(i,j)'$ is used while performing the transition between time j and $j+1$, which also explains why the label is not present for time $p(n)$.

The set of objects of the simulating P system will be:

$$\begin{aligned}\Gamma =& \{a_{i,j,k} \mid i,j \in \{0,\dots,p(n)\},\ 0 \le i < m+5, a \in \Sigma\}\\ &\cup \{q_{i,j,k} \mid i,j \in \{0,\dots,p(n)\},\ 0 \le i \le m+5, q \in Q\}\\ &\cup \{q_{i,j,k,a} \mid i,j \in \{0,\dots,p(n)\},\ 0 \le i \le m+5,\ q \in Q, a \in \Sigma\}\\ &\cup \{a_i \mid a \in \Sigma, i \in \{0,\dots,p(n)\}\} \cup \{q^I\}\end{aligned}$$

where $m = |\Sigma|$ and q^I is the initial state of the TM. The first three sets of the union represent, respectively, the symbols on the tape, the state of the TM, and the state of the TM together with the symbol present under the tape head. The last two sets are only used to encode the initial configuration of the TM. Let $a_1, a_2, \dots, a_{p(n)}$ be the initial contents of the TM tape. It is encoded in the initial configuration of Π as the objects $a_{1,1}, a_{2,2}, \dots, a_{p(n),p(n)}$ inside the outermost membrane (e.g., if the initial content of the tape is *abba*, then it will be encoded by the multiset $a_1 b_2 b_3 a_4$. The initial state q^I is encoded as the object q^I.

The rules of the P system performing the simulation of the TM M are presented both in the main text and grouped together in Fig. 1. The following rules send the objects representing the TM tape inside the corresponding membranes: the object a_i is sent into the membrane $(i,0)$. At the same time the object q^I is rewritten as $q^I_{0,0,0}$:

$$a_i\ [\,]_{(i,0)} \to [a_{i,0,0}]_{(i,0)} \qquad \text{for } a \in \Sigma \tag{1}$$

$$[q^I \to q^I_{0,0,0}]_0 \tag{2}$$

After this first "bookkeeping" step, the actual simulation of one TM step can start. The previous rules will not be further applied during the simulation. An example of simulation of one step of M by Π is presented in Fig. 2.

Let φ be a bijection from Σ to $\{1,\dots,|\Sigma|\}$ providing a total ordering of the TM alphabet. The main idea is to have each object representing the symbol a written on position i at time j on the TM tape dissolving the membrane (i,j) when its subscript is $i, j, \varphi(a)$. This means that any other object present in the same membrane (in our case, the object representing the current state of the TM) can infer the symbol under the tape head and act accordingly. The evolution of the objects representing the tape content for the first $m+1$ time steps of the simulation of each TM step is described by the following rules:

$$[a_{i,j,k} \to a_{i,j,k+1}]_{(i.j)} \qquad \text{for } 0 \le k < \varphi(a) \text{ and } a \in \Sigma \tag{3}$$

$$[a_{i,j,k}]_{(i,j)} \to a_{i,j,k+1} \qquad \text{for } k = \varphi(a) \text{ and } a \in \Sigma \tag{4}$$

$$[a_{i,j,k} \to a_{i,j,k+1}]_0 \qquad \text{for } \varphi(a) < k \le m \text{ and } a \in \Sigma \tag{5}$$

where k goes from 0 to m and it is used to decide when a membrane has to be dissolved. Notice how the objects simply "count" in the subscript except that when $k = \varphi(a)$ the membrane in which they are contained is dissolved. The application of those rules is depicted in Fig. 2 in the transitions from $t = 1$ to $t = 2$, from $t = 2$ to $t = 3$, and from $t = 3$ to $t = 4$; the dissolution of the

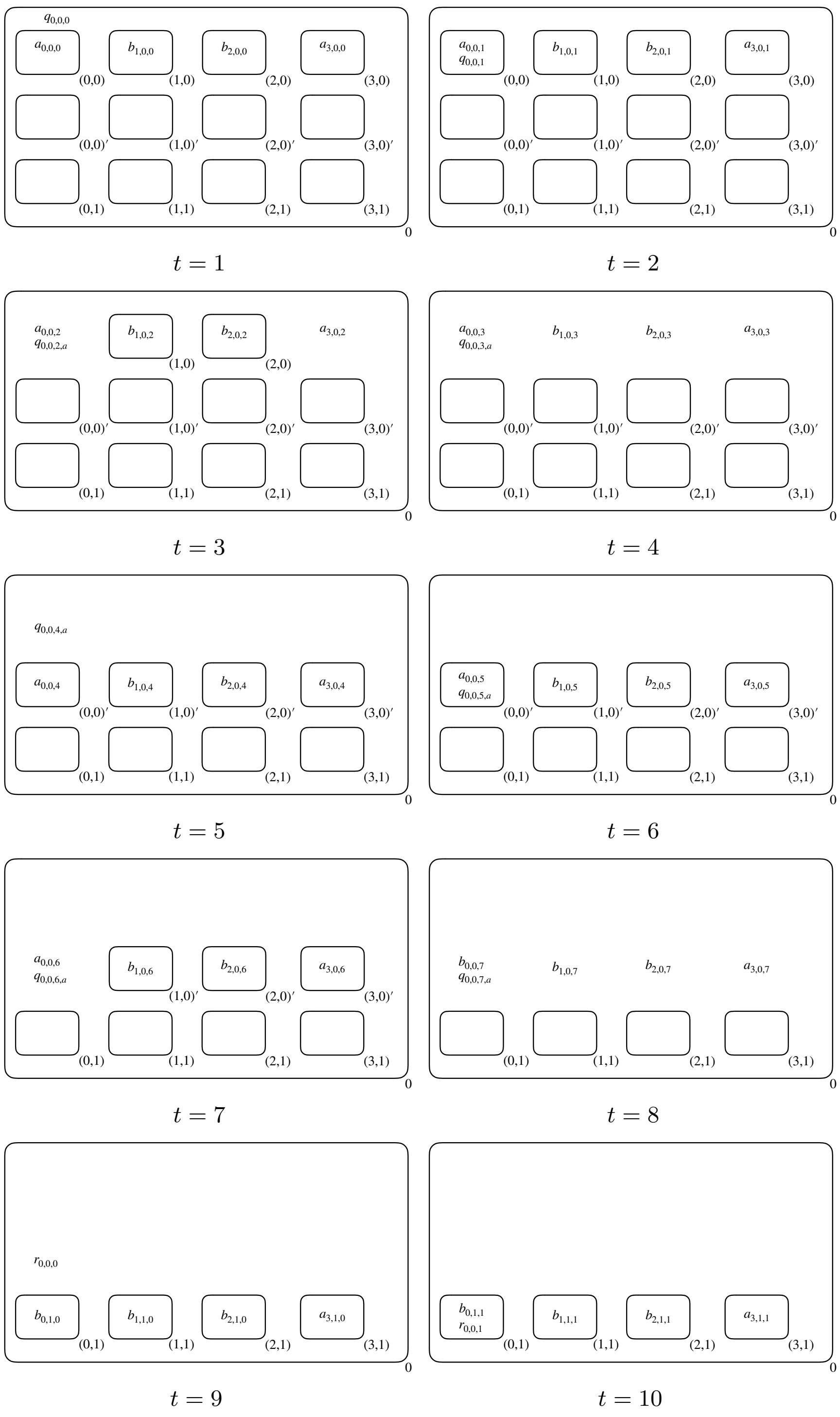

Fig. 2. The simulation of one computation step of the TM M by means of the P system Π. In this example the alphabet Σ is $\{a, b\}$ and the tape contains four cells.

containing membranes happens from $t = 2$ to $t = 3$ for objects representing the tape symbol a, and from $t = 3$ to $t = 4$ for objects representing the tape symbol b.

At the same time the object representing the TM state enters the membrane (i, j), representing that the tape head at time j is in position i and starts to count. When membrane (i, j) is dissolved it is possible to infer the object that dissolved it and, thus, the symbol on the tape under the tape head, which is represented by $\varphi^{-1}(a)$ (which is well defined since φ is a bijection between Σ and $\{1, \ldots, m\}$. The corresponding rules are:

$$q_{i,j,0}\ [\ ]_{(i,j)} \rightarrow [q_{i,j,1}]_{(i,j)} \qquad \text{for } q \in Q \tag{6}$$

$$[q_{i,j,k} \rightarrow q_{i,j,k+1}]_{(i,j)} \qquad \text{for } 1 \leq k \leq m \text{ and } q \in Q \tag{7}$$

$$[q_{i,j,k} \rightarrow q_{i,j,k+1,\varphi^{-1}(k)}]_0 \qquad \text{for } 1 \leq k \leq m, \text{ and } q \in Q \tag{8}$$

$$[q_{i,j,k,a} \rightarrow q_{i,j,k+1,a}]_0 \qquad \text{for } 1 \leq k \leq m,\ a \in \Sigma, \text{ and } q \in Q \tag{9}$$

The application of those rules is depicted in Fig. 2 in all the transitions from $t = 1$ to $t = 4$. At time step $m + 1$ in the simulation of the current TM step, all membranes with label (i, j) (for all i and with j the current TM step being simulated) have been dissolved. Now the object representing the TM state continues to wait in the outermost membrane while *all* the objects representing the TM tape are sent in into the corresponding membranes $(i, j)'$. These membranes will be employed to delete the current content of the cell under the TM head and to replace it with the new symbol. The rules applied at time step $m + 1$ are the following ones:

$$a_{i,j,m+1}\ [\ ]_{(i,j)'} \rightarrow [a_{i,j,m+2}]_{(i,j)'} \qquad \text{for } a \in \Sigma \tag{10}$$

$$[q_{i,j,m+1,a} \rightarrow q_{i,j,m+2,a}]_0 \qquad \text{for } q \in Q \text{ and for } a \in \Sigma \tag{11}$$

In Fig. 2 the application of these rules is in the transition from $t = 4$ to $t = 5$. Once all the objects of the form $a_{i,j,k}$ for $a \in \Sigma$ have entered the membranes $(i, j)'$, they wait for the object representing the TM state to enter:

$$[a_{i,j,m+2} \rightarrow a_{i,j,m+3}]_{(i,j)'} \qquad \text{for } a \in \Sigma \tag{12}$$

$$q_{i,j,m+2,a}\ [\ ]_{(i,j)'} \rightarrow [q_{i,j,m+3,a}]_{(i,j)'} \qquad \text{for } q \in Q \text{ and } a \in \Sigma \tag{13}$$

These rules are applied in the transition from $t = 5$ to $t = 6$ in Fig. 2. At time step $m + 3$ the membrane containing the object representing the TM state is dissolved. In all other membranes the objects representing the TM tape wait for one more step:

$$[a_{i,j,m+3} \rightarrow a_{i,j,m+4}]_{(i,j)'} \qquad \text{for } a \in \Sigma \tag{14}$$

$$[q_{i,j,m+3,a}]_{(i,j)'} \rightarrow q_{i,j,m+4,a} \qquad \text{for } q \in Q \text{ and } a \in \Sigma \tag{15}$$

In Fig. 2 these rules are applied in the transition from $t = 6$ to $t = 7$. One of the focal points of this simulation algorithm happens at time step $m + 4$

(always relative to the start of the simulation of the current TM step). Here, all the objects representing the tape content dissolve the membrane $(i,j)'$ in which they are located. The *only* object not performing this step is the one that was sent into the outermost membrane by the dissolution triggered by the object representing the TM state. The object representing the old content of the tape cell is deleted (by rewriting it to the empty multiset ϵ) and the one encoding the TM state produces its replacement according to the transition function δ of the TM:

$$[a_{i,j,m+4}]_{(i,j)'} \to a_{i,j,m+5} \qquad \text{for } a \in \Sigma \qquad (16)$$

$$[a_{i,j,m+4} \to \epsilon]_0 \qquad \text{for } a \in \Sigma \qquad (17)$$

$$[q_{i,j,m+4,a} \to q_{i,j,m+5,a}\ b_{i+d,j,m+5}]_0 \qquad \text{for } q \in Q,\ a \in \Sigma, \text{ and } \delta(q,a) = (r,b,d) \qquad (18)$$

These rules are applied in the transition from $t = 7$ to $t = 8$ in Fig. 2. Finally, a new simulation step can start by sending in all the objects representing the TM tape to the membranes $(i, j+1)$ and resetting the last component of their subscript. At the same time the object representing the TM state actually applies the transition function δ and rewrites itself:

$$a_{i,j,m+5}\ [\ ]_{(i,j+1)} \to [a_{i,j+1,0}]_{(i,j+1)} \qquad \text{for } a \in \Sigma \qquad (19)$$

$$[q_{i,j,m+5,a} \to r_{i+d,j+1,0}]_0 \qquad \text{for } q \in Q,\ a \in \Sigma, \text{ and } \delta(q,a) = (r,b,d) \qquad (20)$$

The application of these rules happens in the transition from $t = 8$ to $t = 9$ in Fig. 2. The last transition (from $t = 9$ to $t = 10$) is the beginning of the simulation of the next step.

Notice that all rules, labels, and objects can be constructed by a logarithmic space TM. In fact, most of them are constructed by iterating either a constant or a polynomial number of times to produce the necessary subscripts. Since the counters are all at most polynomial in the number that they contain, they can be encoded using a logarithmic number of bits.

We can thus state the main result:

Theorem 1. (L, L)*-uniform families of confluent* shallow *polarizationless P systems with active membranes with dissolution can solve all problems in* P*.*

This result was already known for non-shallow systems [5], but here there are two main innovations: the systems here are *shallow*, i.e., of depth 1, and the construction is via a direct simulation of a Turing machine, which allows one to embed this construction into more complex membrane structures. Furthermore, notice how the construction makes no use of rules duplicating the membranes and their contents (i.e., *division* rules), which are a common way of increasing the computational power of P systems.

Notice that the above construction can be modified to simulate a non-deterministic TM by replacing the only two types of rules involving the transition function of the TM in such a way to allow for a non-deterministic choice (due to having multiple rules in conflict):

$$[q_{i,j,m+4,a} \to q_{i,j,m+5,(r,b,i+d)}\ b_{i+d,j,m+5}]_0 \qquad \text{for } q \in Q,\ a \in \Sigma, \text{ and } (r,b,d) \in \delta(q,a)$$

$$[q_{i,j,m+5,(r,b,i+d)} \to r_{i+d,j+1,0}]_0 \qquad \text{for } q \in Q \text{ and } a \in \Sigma$$

In the first rule the non-deterministic choice is remembered by writing it in the subscript. In this way, the only rule of the second kind that can fire is the one corresponding to the non-deterministic choice performed. We can then state the following theorem showing that a weaker uniformity condition is still sufficient to reach NP for non-confluent systems:

Theorem 2. (L, L)*-uniform families of* non-confluent shallow *polarizationless P systems with active membranes with dissolution and without division can solve all problems in* NP.

4 Conclusions and Open Problems

In this paper we showed that P systems without charges can still solve all problems in the complexity class P even when the power of the machines in the uniformity condition is reduced. The TM simulation presented is quite modular and can be embedded in more complex membrane structures. The resulting simulation is also efficient, requiring a slowdown of only a constant multiplicative factor.

Among the open problems, the most prominent one is to study if the construction presented in [1] can be replicated for systems with charges, possibly adding an additional nesting level to accommodate for the different TM simulation technique. Such a result would show that even without charges the entirety of the counting hierarchy is reachable in constant depth. This is another step in trying to understand what are the features that actually grant P systems the power to reach beyond P and, in some cases, beyond the entire polynomial hierarchy. It would also be interesting to modify the simulation to reduce the number of labels and, consequently, membranes employed: the current construction uses $O(p(n)^2)$ membranes (i.e., one for every pair of tape cell and time step). However, the number of cells that are rewritten during a computation of the TM halting in $p(n)$ steps is at most $p(n)$. Can we reduce the number of membrane labels to this quantity? Finally, we have shown that (L, L)-uniformity is sufficient to construct the P system. We conjecture that this is not necessary and that, in fact, the construction might be possible with even weaker uniformity conditions; it remains open to find what those conditions are.

References

1. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Membrane division, oracles, and the counting hierarchy. Fundamenta Informaticae **138**(1–2), 97–111 (2015), https://doi.org/10.3233/FI-2015-1201
2. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Characterising the complexity of tissue P systems with fission rules. Journal of Computer and System Sciences **90**, 115–128 (2017), https://doi.org/10.1016/j.jcss.2017.06.008
3. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: The counting power of P systems with antimatter. Theoretical Computer Science **701**, 161–173 (2017), https://doi.org/10.1016/j.tcs.2017.03.045
4. Martín-Vide, C., Păun, G., Pazos, J., Rodríguez-Patón, A.: Tissue P systems. Theoretical Computer Science **296**(2), 295–326 (2003), https://doi.org/10.1016/S0304-3975(02)00659-X
5. Murphy, N., Woods, D.: The computational power of membrane systems under tight uniformity conditions. Natural Computing **10**(1), 613–632 (2011), https://doi.org/10.1007/s11047-010-9244-7
6. Păun, G.: Computing with membranes. Journal of Computer and System Sciences **61**(1), 108–143 (2000), https://doi.org/10.1006/jcss.1999.1693
7. Păun, G.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics **6**(1), 75–90 (2001)
8. Păun, G.: Further twenty six open problems in membrane computing. In: Gutíerrez-Naranjo, M.A., Riscos-Nuñez, A., Romero-Campero, F.J., Sburlan, D. (eds.) Proceedings of the Third Brainstorming Week on Membrane Computing. pp. 249–262. Fénix Editora (2005), http://www.gcn.us.es/3BWMC/Volumen.htm
9. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
10. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) Unconventional Models of Computation, UMC'2K, Proceedings of the Second International Conference, pp. 289–301. Springer (2001), https://doi.org/10.1007/978-1-4471-0313-4_21
11. Zandron, C., Leporati, A., Ferretti, C., Mauri, G., Pérez-Jiménez, M.J.: On the computational efficiency of polarizationless recognizer P systems with strong division and dissolution. Fundamenta Informaticae **87**, 79–91 (2008), http://content.iospress.com/articles/fundamenta-informaticae/fi87-1-06