Acta Informatica 41, 111–144 (2004)
Digital Object Identifier (DOI) 10.1007/s00236-004-0153-z

Trading polarizations for labels in P systems with active membranes

Artiom Alhazov[1,2], **Linqiang Pan**[1,3], **Gheorghe Păun**[4,5]

[1] Research Group on Mathematical Linguistics, Rovira i Virgili University, Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain (e-mail: {artiome.alhazov@estudiants, lp@fll}.urv.es)
[2] Institute of Mathematics and Computer Science, Academy of Science of Moldova, Str. Academiei 5, 2028 Chişinău, Moldova (e-mail: artiom@math.md)
[3] Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China (e-mail: lqpan@mail.hust.edu.cn)
[4] Institute of Mathematics of the Romanian Academy, PO Box 1-764, 70700 Bucureşti, Romania (e-mail: george.paun@imar.ro)
[5] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain (e-mail: gpaun@us.es)

Received: 29 October 2003 / 22 September 2004
Published online: 29 October 2004 – © Springer-Verlag 2004

Abstract. This paper addresses the problem of removing the polarization of membranes from P systems with active membranes – and this is achieved by allowing the change of membrane labels by means of communication rules or by membrane dividing rules. As consequences of these results, we obtain the universality of P systems with active membranes which are allowed to change the labels of membranes, but do not use polarizations. Universality results are easily obtained also by direct proofs. By direct constructions, we also prove that SAT can be solved in linear time by systems without polarizations and with label changing possibilities. If non-elementary membranes can be divided, then SAT can be solved in linear time without using polarizations and label changing. Several open problems are also formulated.

1 Introduction

Membrane systems (or P systems) are biologically motivated theoretical models of distributed parallel computing, introduced in [9], which can be seen as a general computing architecture where various types of objects can be processed by various operations. For a motivation and detailed description of various P system models we refer to [9,11].

Very briefly, a P system processes multisets of symbol-objects placed in the compartments of a hierarchical arrangement of cell-like membranes; the external membrane is called the *skin* membrane, while a membrane without any other membrane inside is said to be *elementary*. In each compartment, local evolution rules are used, in the form of multiset rewriting rules. The rules are applied in a maximally parallel way (all objects which can evolve should do it), non-deterministically chosing the rules and the objects. A computation (that is, a sequence of transitions of the system from a configuration to another configuration, by using the evolution rules as mentioned above) is successful only if it halts, reaches a configuration where no rule is applicable. With a halting computation a result is associated, in the form of the multiplicity of objects placed in a specified membrane (internal output) or sent out of the system during the computation (external output). Many types of P systems can simulate in this way any Turing machine (we say that they are computationally universal). When addressing decision problems, a P system is used in the accepting mode: an encoding of the problem is introduced in the initial configuration, and the computation stops by sending to the environment either an object yes or an object no, depending on whether or not the problem has a positive or a negative answer, respectively. We refer to [11–13] etc. for details.

When solving computationally hard problems, the usual approach is the brute-force one, based on a a time-space trade-off. A frequently used way for obtaining an exponential working space in a linear time is membrane division, inspired from cell division well-known in biology. This operation will be also used below. Using it, many hard problems, typically **NP**-complete problems, were shown to be solvable in polynomial (often, linear) time by P systems. Details can be found in [10,11,13]. Recently, also **PSPACE**-complete problems were attacked in this way (see [16,1]).

Informally speaking, in P systems with active membranes one uses six types of rules: (a) multiset rewriting rules, (b) rules for introducing objects into membranes, (c) rules for sending objects out of membranes, (d) rules for dissolving membranes, (e) rules for dividing elementary membranes, and (f) rules for dividing non-elementary membranes. For most of this paper we discuss only rules of the first five types; rules for dividing non-elementary membranes are used only in Section 5.3, in a particular form which we will introduce directly there.

All these rules are associated with membranes, while membranes have both a label, from a finite set of labels, and an "electrical polarization", which can be $+$, $-$, or 0 (for "neutral").

However, the electrical charges used in this context are not quite realistic from a biological point of view. The cell membrane is polarized, but in a different manner: it is in general positively charged in the external layer and negatively charged in the inner layer of phospholypidic molecules. On the other hand, it is possible to have an overall polarization, positive or negative, of inner vesicles, due to ion exchanges with the inner or upper compartments, but the changes of these polarizations are done by more complex processes than by applying single rule as in P systems with active membranes.

Anyway, it is a natural question to consider systems without membrane polarization – this is to say, systems whose membranes always have the same polarization, for instance, they are neutral. What is the power and what is the efficiency of such systems? In particular, are they universal? Can they solve **NP**-complete problems efficiently?

The general questions above remain open and we give here only a partial answer to them. First, we prove that without polarizations we can compute the Parikh sets of matrix languages (generated by grammars without appearance checking), and also the Parikh sets of some non-matrix languages. Then, we address a simpler problem: what else can be added to P systems with active membranes such that by removing the polarizations we still get universality and polynomial solutions to **NP**-complete problems? A suggestion comes already from [11], although not introduced there with this goal: let us allow the membrane division rules to introduce membranes with new labels, not necessarily with the same label as the divided membrane. The idea can be extended also to rules of types (b) and (c): change the label of a membrane when introducing or expelling an object in/from a membrane.

As we shall see below, for rules of types (c) and (e), systems with polarized membranes can be simulated by systems with non-polarized membranes, provided that we can change the labels of membranes when sending objects out of them or when dividing them. For rules of type (e) the result is true for systems with only two levels of membranes which never change the polarization of the skin. Note that this is somewhat similar to the biological observation mentioned above, that the inner vesicles of a cell might be polarized, while, moreover, two levels of membranes is a good approximation of the structure of a cell.

Pleasantly enough, for proving the universality of P systems with active membranes (and polarization), systems of depth two suffice, and they do not change the polarization of the skin. Thus, in both cases, we get the universality as a corollary of the above mentioned results (and the known proof of universality from [6, 11] – slightly modified).

The case of rules of type (b) remains open: can a P system with active membranes and polarizations be simulated by a system without polarizations but allowed to change the label of membranes when introducing objects into them? Even without such a simulation lemma, the universality can be obtained also in this case, by a (surprisingly simple) direct proof.

One already showed that the satisfiability problem of propositional formulae in the conjunctive normal form (`SAT` problem) can be solved in linear time with respect to the number of variables and the number of clauses by a P system with active membranes and polarization using rules of types $(a), (b), (c), (e)$. Unfortunately, the proofs of the two simulation results mentioned above have a drawback: they introduce non-determinism in the functioning of the system. This means that they do not imply that `SAT` can be solved in polynomial time by systems without polarization – but this is directly proven, for the two “easy" cases, of changing labels by rules of type (c), or (e) (and again it remains open for the case of rules of type (b)).

An interesting result is obtained when using rules of type (f), for dividing non-elementary membranes: `SAT` is solved in linear time (by a system constructed in a semi-uniform manner) without using polarizations and label changing (the universality remains open in this case).

2 Prerequisites and definitions

2.1 Elements of formal language theory

We assume the reader to be familiar with basic elements of complexity theory and formal language theory, for instance, from [7, 14, 15]. For an alphabet V, by V^* we denote the free monoid generated by V under the operation of concatenation; the empty string is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. By *REG, CF, CS, RE* we denote the families of languages from Chomsky hierarchy (regular, context-free, context-sensitive, recursively enumerable languages, respectively); for a family FL, by $PsFL$ we denote the family of Parikh sets of languages in FL. As usual, the Parikh mapping associated with an alphabet V is denoted by Ψ_V. By $\mathbb{N}$ we denote the set of non-positive integers. In the following we will not distinguish between a vector $(y_1, \ldots, y_\beta) \in \mathbb{N}^\beta$, its representation by a multiset or its representation by a string with Parikh vector $(y_1, \ldots, y_\beta)$.

Let U be an arbitrary set. A multiset (over U) is a mapping $M : U \to \mathbb{N}$; $M(a)$, for $a \in U$, is the multiplicity of a in the multiset M. We indicate this fact also in the form $(a, M(a))$. If the set U is finite, a multiset M over U, $\{(a_1, M(a_1), \ldots, (a_n, M(a_n)\}$, can also be represented by a string: $w = a_1^{M(a_1)} \ldots a_n^{M(a_n)}$ or by any of its permutations. In the following we will not distinguish the representation of multiset by a mapping or by a string.

A *matrix grammar with appearance checking* is a computationally universal rewriting system essentially used below. Details (about regulated rewriting) can be found in [3]. For each matrix grammar there is an equivalent matrix grammar in the binary normal form (this is true both for grammars with appearance checking and without appearance checking; in the latter case, the set F and the matrices of type 3 described below are missing).

A matrix grammar $G = (N, T, S, M, F)$ is in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, and the matrices in M are in one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$,
3. $(X \to Y, A \to \#)$, with $X, Y \in N_1, A \in N_2$,
4. $(X \to \lambda, A \to x)$, with $X \in N_1, A \in N_2$, and $x \in T^*$, $|x| \leq 2$.

Moreover, there is only one matrix of type 1 (that is why one uses to write it in the form $(S \to X_{init}A_{init})$, in order to fix the symbols X, A present in it), and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3; $\#$ is a trap-symbol, because once introduced, it is never removed. A matrix of type 4 is used only once, in the last step of a derivation.

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $m \in M$ such that applying once each rule of m to w one can obtain z. A rule can be skipped if it is in F and it is not applicable.

The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. If the set F is empty, then the grammar is said to be without appearance checking; the corresponding family of languages is denoted by MAT.

It is known that $CF \subset MAT \subset MAT_{ac} = RE$, $PsREG = PsCF \subset PsMAT \subset PsRE$, and that $CS - MAT \neq \emptyset$, $PsCS - PsMAT \neq \emptyset$ (for instance, the one-letter languages in MAT are known to be regular, [5]).

In the proofs below we will use the following slight variant of the binary normal form: we say that $G = (N, T, S, M, F)$ is in the *f-binary normal form* if $N = N_1 \cup N_2 \cup \{S, f, \#\}$, and instead of matrices $(X \to \lambda, A \to x)$ of type 4 we have matrices of the forms $(4')$ $(X \to f, A \to x)$ and $(4'')$ $(f \to \lambda)$; the new nonterminal f appears only in these matrices and $(f \to \lambda)$ is the only terminal matrix.

2.2 P systems with active membranes

We also assume the reader to be familiar with the basic knowledge of membrane computing, in particular, with P systems with active membranes, for instance, from [11] (details and recent results from membrane computing can be found at the web address http://psystems.disco.unimib.it).

A P system *with active membranes* (and electrical charges) is a construct

$$\Pi = (O, H, \mu, w_1, \ldots, w_m, R),$$

where:

1. $m \geq 1$ (the initial degree of the system);
2. O is the alphabet of *objects*;
3. H is a finite set of *labels* for membranes;
4. μ is a *membrane structure*, consisting of m membranes with initially neutral polarizations, labeled (not necessarily in a one-to-one manner) with elements of H;
5. $w_1, \ldots, w_m$ are strings over O, describing the *multisets of objects* (every symbol in a string represents one copy of the corresponding object) placed in the m regions of μ;
6. R is a finite set of *developmental rules*, of the following forms:
   (a) $[\, a \to v]_h^e$,
   for $h \in H, e \in \{+, -, 0\}, a \in O, v \in O^*$
   (object evolution rules, associated with membranes and depending on the label and the charge of the membranes, but not directly involving the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them);
   (b) $a[\ ]_h^{e_1} \to [\, b]_h^{e_2}$,
   for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$
   (communication rules; an object is introduced in the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);
   (c) $[\, a\,]_h^{e_1} \to [\ ]_h^{e_2} b$,
   for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$
   (communication rules; an object is sent out of the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);
   (d) $[\, a\,]_h^e \to b$,
   for $h \in H, e \in \{+, -, 0\}, a, b \in O$
   (dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);
   (e) $[\, a\,]_h^{e_1} \to [\, b\,]_h^{e_2} [\, c\,]_h^{e_3}$,
   for $h \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O$
   (division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects; the remaining objects are duplicated and may evolve in the same step by rules of type (a)).

The rules of type (a) are applied in the parallel way (all objects which can evolve by such a rule should do it), while the rules of types (b), (c), (d), (e) are used sequentially, in the sense that one membrane can be used by at most one rule of these types at a time. In total, the rules are used in the non-deterministic maximally parallel manner: all objects and all membranes which can evolve, should evolve. Only halting computations give a result, and the result is the vector of natural numbers describing the multiplicity of objects expelled into the environment during the computation; the set of vectors computed in this way by the various halting computations in Π is denoted by $Ps(\Pi)$. (Non-halting computations give no result.) Details can be found in [11].

A P system is called *deterministic* if there is a single computation. A P system is called *confluent* if either all of its computations do not halt, or all of its computations reach the same halting configuration.

2.3 Removing polarizations, changing labels

Let us introduce now rules – of types $(a) - (e)$ – without polarization. They are of the following forms (because “no polarization" means “neutral polarization", we add the subscript 0 to the previous letters identifying the five types of rules; as above, O is the alphabet of objects and H is the set of labels of membranes):

(a_0) $[\, a \to v]_h$, where $a \in O, v \in O^*$, and $h \in H$,
(b_0) $a[\,]_h \to [\, b]_h$, where $a, b \in O$ and $h \in H$,
(c_0) $[\, a]_h \to [\,]_h b$, where $a, b \in O$ and $h \in H$,
(d_0) $[\, a]_h \to b$, where $a, b \in O$ and $h \in H$,
(e_0) $[\, a]_h \to [\, b]_h [\, c]_h$, where $a, b, c \in O$ and $h \in H$.

Rules of types $(a), (b), (c)$ were introduced without the capability of changing the label of membranes they involve (this makes no sense for dissolving rules), but in [11] one already considers rules of type (e) which can change both the label and the polarization of membranes. Such rules are of the form

$$[\, a]_{h_1}^{e_1} \to [\, b]_{h_2}^{e_2} [\, c]_{h_3}^{e_3},$$
$$\text{with } a, b, c \in O,\ e_1, e_2, e_3 \in \{+, -, 0\},\ \text{and } h_1, h_2, h_3 \in H,$$

and they have been called of type (e'). We extend this idea and this notation to rules of types (e_0), (b_0), and (c_0): their primed versions indicate the fact that the labels can be changed. Specifically, these rules are of the following forms:

(b_0') $a[\,]_{h_1} \to [\, b]_{h_2}$, where $a, b \in O$ and $h_1, h_2 \in H$,

(c_0') $[\,a\,]_{h_1} \to [\,]_{h_2} b$, where $a, b \in O$ and $h_1, h_2 \in H$,
(e_0') $[\,a\,]_{h_1} \to [\,b\,]_{h_2} [\,c\,]_{h_3}$, where $a, b, c \in O$ and $h_1, h_2, h_3 \in H$.

By $PsOP(r)$ we denote the family of sets $Ps(\Pi)$ computed as described above by P systems using rules of types listed in r.

3 The power of P systems without polarizations

The proof of the following result can be found in [11]. Although in [11] this result is given for sets of natural numbers, hence one-dimensional vectors, the extension to vectors of arbitrary dimensions is obvious and it holds with the same proof.

Theorem 1. $PsOP(a, b, c) = PsRE$.

What is the power of P systems without polarizations? Specifically: what is the size of the family $PsOP(a_0, b_0, c_0, d_0, e_0)$? Is it equal to $PsRE$? If not (as we expect), then what about its relation with $PsET0L$ (the Parikh sets of ET0L languages)? We leave these problems open, and we only prove here that we can cover in this way at least the Parikh sets of languages generated by matrix grammars without appearance checking.

The following result is non-trivial, because $PsMAT - PsCF \neq \emptyset$ (there are non-semilinear sets of vectors in $PsMAT$, which is not the case with $PsCF$), but gives only a partial answer to the question how powerful P systems without polarization are.

Theorem 2. $PsMAT \underset{\neq}{\subset} PsOP(a_0, b_0, c_0, d_0, e_0)$.

Proof. Let us consider a matrix grammar (without appearance checking) $G = (N, T, S, M)$ in the f-binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, f\}$, and with matrices of the types 1, 2, $4'$, $4''$ specified above. We assume all matrices of types 2 and $4'$ labeled in a one-to-one manner with $m_2, m_3, \ldots, m_t$, for $t-1$ being the number of these matrices.

We construct the P system

$$\Pi = (O, \{1, 2\}, [\,[\ \ ]_2]_1, \bar{X}_{init}\bar{A}_{init}, c, R),$$

where

$$\begin{aligned} O = {} & N_1 \cup N_2 \cup T \cup \{\bar{X} \mid X \in N_1\} \cup \{\bar{A}, A', A'' \mid A \in N_2\} \\ & \cup \{X_{i,j}, X_i \mid X \in N_1, 1 \leq i \leq t+1, -1 \leq j \leq t\} \\ & \cup \{A_{i,j}, A_i, A_i' \mid A \in N_2, 1 \leq i \leq t+1, 0 \leq j \leq t\} \\ & \cup \{c, c_1, c_2, c_2', f, \bar{f}, \#\} \cup \{d_i \mid 0 \leq i \leq t+1\}, \end{aligned}$$

and the set R contains the following rules (together with the rules we give explanations about their use and the functioning of the system Π; in the rules below, we use the morphisms $h_j, 1 \leq j \leq t+1$, defined by $h_j(A) = A_j$ for all $A \in N_2$, and $h_j(a) = a$ for all $a \in T$):

1. $[\bar{X} \to X]_1$, for all $X \in N_1$; $[\bar{A} \to A]_1$, for all $A \in N_2$;
   $[\,c]_2 \to [\,c_1]_2[\,c_2]_2$; $[\,c \to \#]_2$.

While all symbols from the skin region lose the bars, the inner membrane is divided into two membranes with the same label 2 and containing the auxiliary symbols c_1, c_2, respectively. In the membrane containing c_1 we will simulate a matrix of G; at the end of the simulation, this membrane will be dissolved, and the second membrane will be used as starting membrane with label 2, for a further division.

2. $X[\;]_2 \to [\;X_{i,-1}]_2$, for some $m_i : (X \to Y, A \to x) \in M$, $2 \leq i \leq t$;
   $[\,X \to \#]_1$, for all $X \in N_1$; $[\,c_2]_2 \to [\;]_2 c_2'$;
   $[\,c_2 \to \#]_2$; $[\,A \to A']_1$, for all $A \in N_2$.

In the second step of the computation, all symbols $A \in N_2$ get primed, while the unique symbol from N_1 should enter the membrane with label 2 having inside the object c_1 – otherwise, either X introduces the trap-object $\#$ in the skin region, or, if it enters the membrane containing the object c_2, this object cannot exit the membrane and introduces $\#$.

3. $[\,X_{i,-1} \to X_{i,0}]_2$, for all $X \in N_1, 2 \leq i \leq t$;
   $[\,A' \to A'']_1$, for all $A \in N_2$;
   $B'[\;]_2 \to [\;B_{j,0}]_2$, for some $m_j : (Z \to U, B \to y) \in M$, $2 \leq j \leq t$;
   $[\,c_2' \to \#]_1$; $c_2'[\;]_2 \to [\,d_0]_2$.

In the third step, also a symbol $B \in N_2$ can enter into membrane 2 which contains c_1; we will see below that this is obligatory, otherwise the symbol X – with subscripts – already present there will introduce $\#$. At the same time, all symbols from the skin get double primed and c_2' enters the second membrane with label 2 transformed into d_0; we will also see below that if c_2' enters the first membrane with label 2, then the computation will never halt.

4. $[\,X_{i,0} \to X_{i,1}]_2$, for all $X \in N_1, 2 \leq i \leq t$;
   $[\,B_{j,0} \to B_{j,1}]_2$, for all $B \in N_2, 2 \leq j \leq t$;
   $[\,d_0 \to d_1]_2$; $[\,A'' \to A_1]_1$, for all $A \in N_2$.

All symbols “start to count", from 1 to $t+1$. Note that during this counting no symbol from the skin membrane can enter any of the inner membranes.

5. $[\,X_{i,k} \to X_{i,k+1}]_2$, for all $X \in N_1, 2 \leq i \leq t, 1 \leq k \leq i-1$;
   $[\,B_{j,k} \to B_{j,k+1}]_2$, for all $B \in N_2, 2 \leq j \leq t, 1 \leq k \leq j-2$;

$[\, d_k \to d_{k+1}]_2$, for all $1 \leq k \leq t$;
$[\, A_k \to A_{k+1}]_1$, for all $A \in N_2, 1 \leq k \leq t$.

These rules are used during the counting.

6. $[\, B_{j,j-1}]_2 \to B'_j$, for all $B \in N_2, 2 \leq j \leq t$;
   $[\, X_{i,i} \to \#]_2$, for all $X \in N_1, 2 \leq i \leq t$.

Only the symbol from N_2 can dissolve membrane 2, and this happens in the moment when it is going to get a subscript j, k with $j = k$. If $i < j$, then X gets the subscript i, i before dissolving the membrane, hence the trap-symbol is introduced. This will also happen if no symbol from N_2 were present in this membrane. If, however, $j < i$, then the membrane will be “prematurely dissolved", and a symbol $X_{i,k}$ with $i > k$ will be left free in the skin region, introducing $\#$ there – see the rules below. In this way we check that the symbols $X_{i,-1}, B_{j,0}$ introduced in the membrane have $i = j$, hence correspond to the same matrix of G.

7. $[\, X_{i,k} \to \#]_1$, for all $X \in N_1, 2 \leq i \leq t, 1 \leq k \leq i-1$;
   $[\, d_k \to \#]_1$, for all $k \leq t$; $\quad [\, c_1 \to \lambda]_1$; $\quad [\, X_{i,i} \to Y_{i+1}]_1$;
   $[\, B'_i \to h_{i+1}(x)]_1$, for $m_i : (X \to Y, B \to x), Y \in N_1 \cup \{f\}$.

The first rule ensures the correct simulation of the matrix m_i, the second one ensures the fact that in the third step d_0 was sent to the second membrane with label 2, while the last two rules actually simulate the matrix. Note that in the meantime, all symbols from N_2 present in the skin region have continued to increase their subscript – and the same with d_k from the second membrane 2. The symbol Y and the nonterminals from x are introduced with the subscript equal to the subscripts of all these symbols from the skin membrane, in order to continue together to count to $t+1$; this is important for the synchronization of the system.

8. $[\, Y_{t+1} \to \bar{Y}]_1$, for all $Y \in N_1 \cup \{f\}$;
   $[\, A_{t+1} \to \bar{A}]_1$, for all $A \in N_2$; $\quad [\, d_{t+1} \to c]_2$.

When all symbols get the subscript $t+1$, we can return to a configuration similar to the initial one, with all nonterminals barred, and with c in membrane 2, hence we can iterate the process, and simulate another matrix of G.

9. $\bar{f}[\,]_2 \to [\,\bar{f}]_2$; $\quad [\,\bar{f}]_2 \to \bar{f}$;
   $[\, A_{i,k} \to \#]_1$, for all $A \in N_2, 1 \leq k < i \leq t$;
   $[\, d_{t+1} \to \lambda]_2$; $\quad [\, \# \to \#]_s$, $s = 1, 2$.

After using the terminal matrix of G, we also have to remove the symbols d and c, and, furthermore, no symbol from N_2 should be present in the system. Assume that this is not the case, hence we have at least $\bar{f}, \bar{A}$ in the skin membrane, for some $A \in N_2$, and c in membrane 2. If $\bar{f}$ enters membrane 2, then c introduces $\#$, hence membrane 2 should be divided, while $\bar{f}$ waits and $\bar{A}$ is replaced by A. In the next step $\bar{f}$ enters the first membrane 2 – the second one should be used by c_2, as we have seen above – and A becomes A'. Now, if $\bar{f}$ dissolves the first membrane 2, then A' should pass to A'' and then start to count; if A' enters membrane 2, then $\bar{f}$ waits, but it dissolves the membrane in the next step, and $A_{i,1}$ is released into the skin region, where it will introduce $\#$. If $\bar{f}$ enters the second membrane 2 and dissolves it, then a symbol d_k is released, and again we introduce $\#$ in the skin region. If no membrane is present, then A will count forever from 1 to $t+1$, repeatedly. Thus, the only way to stop is to correctly simulate a terminal derivation in G, removing d_{t+1} at the same time when introducing $\bar{f}$.

10. $[\,a\,]_1 \to [\,]_1 a$, for all $a \in T$.

Any terminal symbol is sent out at any time of the computation.

From the previous explanations it is easy to see that $\Psi_T(L(G)) = Ps(\Pi)$, which proves the inclusion $PsMAT \subseteq PsOP(a_0, b_0, c_0, d_0, e_0)$ (note that rules of all five types were used).

In order to see that the inclusion is proper, consider the system

$$\Pi = (\{a\}, \{1,2\}, [\,[\,]_2]_1, \lambda, a, \{[\,a \to aa]_2, [\,a]_2 \to a, [\,a]_1 \to [\,]_1 a\}),$$

which generates the set $Ps(\Pi) = \{(2^n - 1) \mid n \geq 1\} \notin PsMAT$. $\square$

We do not know whether the previous result can be improved by avoiding the use of some types of rules, or – more interesting – strengthening it to a characterization of $PsRE$. Because we believe that such a characterization is not possible, a related question is to consider further ingredients which can increase the power. A possibility is to use a priority relation among rules, of a weak type (a general priority is known to lead to universality, see [11]); for example, we can use always the rules of some type with priority over the rules of another type (e.g., always the communication having priority on dividing a membrane). This possibility, as well as the general problem concerning the size of the family $PsOP(a_0, b_0, c_0, d_0, e_0)$, remains open, and we consider in the next section another way to "pay" for not using polarization: changing the labels of membranes.

4 The power of changing labels

Clearly, rules of type (α) or (α_0') are stronger than rules of type (α_0), where $\alpha \in \{b, c, e\}$. It remains open whether these relations ("stronger/weaker") are proper. Here we are going to present two simulation results.

4.1 Simulation results

We start by considering the case when rules of type (e_0') are used. They help at least for a class of systems of a restricted type. Specifically, we say that a P system (with active membranes and using polarization) is *of type D2S0* if its membrane structure has only two levels (depth two, hence D2) and its skin membrane never changes the polarization (hence it remains neutral as at the beginning, a fact indicated by S0).

Lemma 3. *Any P system of type D2S0 can be simulated by a system using rules of types* $(a_0), (b_0), (c_0), (d_0), (e_0')$.

Proof. Let us consider a system $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$ of type D2S0. We assume that the skin membrane is labeled with 1 and that this label is never used for another membrane (this can be easily achieved, by relabeling the membranes, taking into account that the skin membrane is never divided). We also assume the rules of type (b) from R (if any) labeled in a one-to-one manner with elements of a set B (hence we write these rules in the form $r : a[\ ]_h^{e_1} \to [\ b]_h^{e_2}$, $r \in B$). Consider also the alphabet $O_1 = \{a_1 \mid a \in O\}$ and the morphism $\varphi : O^* \longrightarrow O_1^*$ defined by $\varphi(a) = a_1,\ a \in O$.

We construct the system (without polarizations)

$$\Pi' = (O', H', \mu', w_1, \ldots, w_m, R'),$$

with the following components:

$$\begin{aligned} O' &= O \cup \{a_i, a_i', a_i'' \mid a \in O, 1 \le i \le 5\} \\ &\quad \cup \{b_2^{(r)} \mid r : a[\ ]_h^{e_1} \to [\ b]_h^{e_2} \text{ is a rule of type } (b) \text{ from } R, r \in B\} \\ &\quad \cup \{d, \$, \$', \$'', \#\}, \\ H' &= \{\langle h, e\rangle, \langle h', e\rangle \mid h \in H, e \in \{+, -, 0\}\} \cup \{0\}, \end{aligned}$$

μ' is the membrane structure μ with each membrane with label h (and polarization 0, as it is the case at the beginning) labeled with $\langle h, 0\rangle$,

and with the set R' constructed as follows.

- For each rule $[\ a \to x]_h^e \in R$ of type (a), we introduce in R' the rules:

A1 $[\ a \to \varphi(x)]_{\langle h,e\rangle}$;
A2 $[\ b_1 \to b_1']_{\langle h,e\rangle}$; $[\ b_1 \to b_1']_{\langle h',e\rangle}$;
A3 $[\ b_1' \to b_1'']_{\langle h,e\rangle}$; $[\ b_1' \to b_1'']_{\langle h',e\rangle}$;
A4 $[\ b_1'' \to b]_{\langle h,e\rangle}$, $[\ b_1'' \to b]_{\langle h',e\rangle}$, for all $b \in O$.

- For each rule $r : a[\ ]_h^{e_1} \to [\, b]_h^{e_2} \in R$ of type (b), we introduce in R' the rules:

B1 $a[\ ]_{\langle h,e_1\rangle} \to [\, b_2^{(r)}]_{\langle h,e_1\rangle}$;
B2 $[\, b_2^{(r)}]_{\langle h,e_1\rangle} \to [\, b_2]_{\langle h',e_2\rangle}[\, d]_0$; $[\, b_2^{(r)} \to \#]_{\langle h,e_1\rangle}$;
B3 $[\, b_2 \to b_2'\$']_{\langle h',e_2\rangle}$;
B4 $[\, b_2' \to b]_{\langle h',e_2\rangle}$; $[\, \$']_{\langle h',e_2\rangle} \to [\, \$'']_{\langle h,e_2\rangle}[\, d]_0$.

- For each rule $[\, a]_h^{e_1} \to [\ ]_h^{e_2} b \in R$ of type (c), with $h \neq 1$, we introduce in R' the rules:

C1 $[\, a]_{\langle h,e_1\rangle} \to [\, b_3]_{\langle h',e_2\rangle}[\, d]_0$;
C2 $[\, b_3 \to b_3'\$]_{\langle h',e_2\rangle}$;
C3 $[\, b_3']_{\langle h',e_2\rangle} \to [\ ]_{\langle h',e_2\rangle} b_3''$; $[\, \$ \to \$']_{\langle h',e_2\rangle}$;
C4 $[\, b_3'' \to b]_g$, for all $g \in H'$; $[\, \$']_{\langle h',e_2\rangle} \to [\, \$'']_{\langle h,e_2\rangle}[\, d]_0$.

- For each rule $[\, a]_h^{e} \to b \in R$ of type (d), we introduce in R' the rules:

D1 $[\, a]_{\langle h,e\rangle} \to [\, b_4]_{\langle h',e\rangle}[\, d]_0$;
D2 $[\, b_4 \to b_4']_{\langle h',e\rangle}$;
D3 $[\, b_4' \to b_4'']_{\langle h',e\rangle}$;
D4 $[\, b_4'']_{\langle h',e\rangle} \to b$.

- For each rule $[\, a]_h^{e_1} \to [\, b]_h^{e_2}[\, c]_h^{e_3} \in R$ of type (e), we introduce in R' the rules:

E1 $[\, a]_{\langle h,e_1\rangle} \to [\, b_5]_{\langle h',e_2\rangle}[\, c_5]_{\langle h',e_3\rangle}$;
E2 $[\, \alpha_5 \to \alpha_5'\$]_{\langle h',e\rangle}$, for all $\alpha \in O, e \in \{+,-,0\}$;
E3 $[\, \alpha_5' \to \alpha_5'']_{\langle h',e\rangle}$, for all $\alpha \in O, e \in \{+,-,0\}$;
$[\, \$ \to \$']_{\langle h',e\rangle}$, for all $e \in \{+,-,0\}$;
E4 $[\, \alpha_5'' \to \alpha]_{\langle h',e\rangle}$, for all $\alpha \in O, e \in \{+,-,0\}$;
$[\, \$']_{\langle h',e\rangle} \to [\, \$'']_{\langle h,e\rangle}[\, d]_0$, for all $e \in \{+,-,0\}$.

- Finally, for each rule $[\, a]_1^0 \to [\ ]_1^0 b$ of R we introduce in R' the rule
$[\, a]_{\langle 1,0\rangle} \to [\ ]_{\langle 1,0\rangle} b$,
- and then, for all labels $g \in H'$ we introduce the rule
$[\, \# \to \#]_g$.

The idea behind this construction should be visible: instead of working with membranes with labels and polarization, $[\ ]_h^e$, we work with membranes having only labels, $[\ ]_{\langle h,e\rangle}$, with the polarizations “stored" as the second component of the labels; by handling labels we can then handle polarizations; the problem is that the labels are changed only by rules of type (e_0'); moreover, we have to carefully arrange the computations in Π' in order not to lose the synchronization of computations in Π.

The synchronization is obtained by using symbols with subscripts 1, 2, 3, 4, 5, associated with rules of R of the types $(a), (b), (c), (d), (e)$, respectively, sometimes priming these symbols, and also using labels not only of the form $\langle h, e\rangle$, but also of the form $\langle h', e\rangle$. Each step of a computation in Π is simulated by four steps of a computation in Π'. In the first step all symbols are like in O, without subscripts or primes, and the labels of membranes are of the form $\langle h, e\rangle$. After the first step, all objects which can evolve by a rule in R can also evolve by a rule in R' and the objects introduced in this way have subscripts; these objects have now to evolve in a well determined manner, completing the simulation of the rule in R and returning to objects from O only in the last step, the fourth one. Moreover, the simulation of rules of types $(c), (d), (e)$ starts by using a rule of R' of type (e_0'), which introduces a “main membrane" with the label of the type $\langle h', e\rangle$ (with the label $h \in H$ primed), corresponding to the membrane $[\ ]_h^e$ from μ whose rule is simulated, as well as a “dummy membrane", $[\,d]_0$, containing the “dummy object" d which never evolves (no rule is associated with this membrane).

In the simulation of a rule of type (b) one introduces a label of type $\langle h', e\rangle$ in the second step. In the simulation of rules of all types $(b), (c), (e)$ a further division is performed in the fourth step, returning the label $\langle h', e\rangle$ to $\langle h, e\rangle$, thus making possible the simulation of another rule from R.

Because the case of rules of type (b) is different from the case of the other rules, we describe it in some details. Assume that we have a membrane $[\ ]_h^{e_1}$ where we want to use a rule $r : a[\ ]_h^{e_1} \to [\ b]_h^{e_2}$. We start by using the rule $a[\ ]_{\langle h,e_1\rangle} \to [\ b_2^{(r)}]_{\langle h,e_1\rangle}$. This is possible, as the symbol a and the label $\langle h, e_1\rangle$ are available. (In parallel with the rule r, no further rule of types $(b), (c), (d), (e)$ can involve the same membrane, but a maximal use of rules of type (a) should be executed; this is clearly possible also in Π', because the simulation of these rules can also continue in membranes with labels $\langle h', e\rangle$.) The label of the membrane was not changed, but the symbol b got both the subscript 2 (as associated with rules of type (b)) and the superscript (r), to “remember" which rule we have to simulate. In the next step, because the label of the membrane is the same as in the first step, we can involve this membrane in rules of types $(b), (c), (d), (e)$, and this would be wrong, because it does not correspond to a correct simulation of rules from R (the simulation of the rule r was not completed). This is prevented by the rule $[\,b_2^{(r)} \to \#]_{\langle h,e_1\rangle}$: because of the maximal parallelism, it has to be applied, and this will lead to a non-halting computation. Therefore, we have to use the rule $[\,b_2^{(r)}]_{\langle h,e_1\rangle} \to [\,b_2]_{\langle h',e_2\rangle}[\,d]_0$, which continues the correct simulation: the label was changed to $\langle h', e_2\rangle$, hence no new simulation can be started in this membrane (while the rules of R' corresponding to rules of type (a) from R can be applied also in the presence of this label). We continue in a deterministic way with the rule $[\,b_2 \to b_2'\$']_{\langle h',e_2\rangle}$ (step 3 of the simulation), and we conclude by using the rules $[\,b_2' \to b]_{\langle h',e_2\rangle}$, $[\,\$']_{\langle h',e_2\rangle} \to [\,\$'']_{\langle h,e_2\rangle}[\,d]_0$, in parallel. We have returned to a configuration as that we have started with, hence the simulation of rules from R can continue. Note the role of the superscript (r) in step 2, when it was necessary to know the new polarization e_2 of the membrane, as well as the role of the special object \$, primed or not, which takes care of returning the membrane to a label $\langle h, e\rangle$ in the fourth step of the simulation.

The simulation of rules of other types than (b) is easier, in the sense that it is deterministic, no trap-object is used in order to avoid “wrong" simulations.

In any moment, the objects which were sent out of the system by rules of R are also sent out of μ' by the rules of R'.

In all this construction, it is crucial that the skin membrane never changes its polarization (we cannot divide the skin, hence we cannot handle such a case in this framework), and that we have inside the skin membrane only elementary membranes (neither the change of polarization of a non-elementary membrane can be handled, because we cannot divide non-elementary membranes). With these observations, we conclude that the statement in the lemma holds. □

In the previous construction the number of membranes with label 0 can grow arbitrarily large. We can prevent this by introducing the following rules in R':

$$[\,d \to d']_0;\ [\,a \to \lambda]_0, \text{ for all } a \in O';\ [\,d']_0 \to d'';\ [\,d'' \to \lambda]_1.$$

In this way, all objects from membranes with label 0 are removed, in parallel with changing d to d', in the next step the membrane is dissolved, and after that the new object d'' is erased.

We now pass to the case where we can change the label of membranes by means of rules of type (c).

Lemma 4. *Any P system with rules of types* $(a), (b), (c), (d), (e)$ *can be simulated by a system using rules of types* $(a_0), (b_0), (c_0'), (d_0), (e_0)$.

Proof. We start again from a system $\Pi = (O, H, \mu, w_1, \dots, w_m, R)$. Without loss of generality we assume that no membrane has the label s. We also assume all the rules from R labeled in a one-to-one manner with elements of a set B. Consider again the alphabet $O_1 = \{a_1 \mid a \in O\}$ and the morphism $\varphi : O^* \longrightarrow O_1^*$ defined by $\varphi(a) = a_1,\ a \in O$.

We construct the system (without polarizations)

$$\Pi' = (O', H', \mu', w_1, \ldots, w_m, R'),$$

with the following components:

$O' = O \cup \{a', a_1, a'_1, a''_1, a'''_1 \mid a \in O\}$
$\quad \cup \{a^{(r)}, a'^{(r)}, a''^{(r)}, a'''^{(r)} \mid a \in O, r \in B\}$
$\quad \cup \{a_e \mid a \in O, e \in \{+,-,0\}\} \cup \{\$_e \mid e \in \{+,-,0\}\} \cup \{\$, \$', \#\}$,
$H' = \{\langle h, e\rangle, \langle h, e, r\rangle \mid h \in H, e \in \{+,-,0\}, r \in B\} \cup \{s\}$,
μ' is obtained from the membrane structure μ by using
a new membrane, with label s, to enclose the membrane structure μ
(this is the skin membrane of μ') and each membrane in μ
with label h being labeled with $\langle h, 0\rangle$,

and with the set R' constructed as follows.

- For each rule $[\, a \to x]_h^e \in R$ of type (a), we introduce in R' the rules:

A1 $[\, a \to \varphi(x)]_{\langle h,e\rangle}$;
A2 $[\, b_1 \to b'_1]_{\langle h,e\rangle}$; $[\, b_1 \to b'_1]_{\langle h,e,r\rangle}$;
A3 $[\, b'_1 \to b''_1]_{\langle h,e\rangle}$; $[\, b'_1 \to b''_1]_{\langle h,e,r\rangle}$;
A4 $[\, b''_1 \to b'''_1]_{\langle h,e\rangle}$; $[\, b''_1 \to b'''_1]_{\langle h,e,r\rangle}$;
A5 $[\, b'''_1 \to b]_{\langle h,e\rangle}$, $[\, b'''_1 \to b]_{\langle h,e,r\rangle}$, for all $b \in O$, and $r \in B$.

- For each rule $r : a[\ ]_h^{e_1} \to [\, b]_h^{e_2} \in R$ of type (b), we introduce in R' the rules:

B1 $a[\ ]_{\langle h,e_1\rangle} \to [\, b^{(r)}]_{\langle h,e_1\rangle}$;
B2 $[\, b^{(r)}]_{\langle h,e_1\rangle} \to [\ ]_{\langle h,e_2,r\rangle} b'^{(r)}$;
$[\, b^{(r)} \to \#]_g$, for all $g \in \{\langle h,e\rangle \mid h \in H, e \in \{+,-,0\}\}$;
B3 $b'^{(r)}[\ ]_{\langle h,e_2,r\rangle} \to [\, b^{(r)}]_{\langle h,e_2,r\rangle}$;
B4 $[\, b^{(r)} \to b'\$]_{\langle h,e_2,r\rangle}$;
B5 $[\, b' \to b]_{\langle h,e_2,r\rangle}$; $[\, \$]_{\langle h,e_2,r\rangle} \to [\ ]_{\langle h,e_2\rangle} \$'$.

- For each rule $r : [\, a]_h^{e_1} \to [\ ]_h^{e_2} b \in R$ of type (c), we introduce in R' the rules:

C1 $[\, a]_{\langle h,e_1\rangle} \to [\ ]_{\langle h,e_2,r\rangle} b'^{(r)}$;
C2 $b'^{(r)}[\ ]_{\langle h,e_2,r\rangle} \to [\, b^{(r)}]_{\langle h,e_2,r\rangle}$;

C3 $[\, b^{(r)} \to b''^{(r)}]_{\langle h, e_2, r\rangle}$;
C4 $[\, b''^{(r)} \to b'''(r)]_{\langle h, e_2, r\rangle}$;
C5 $[\, b'''(r)]_{\langle h, e_2, r\rangle} \to [\ ]_{\langle h, e_2\rangle} b$.

- For each rule $r : [\, a]^{e}_{h} \to b \in R$ of type (d), we introduce in R' the rules:

D1 $[\, a]_{\langle h,e\rangle} \to [\ ]_{\langle h,e,r\rangle} b'^{(r)}$;
D2 $b'^{(r)}[\ ]_{\langle h,e,r\rangle} \to [\, b^{(r)}]_{\langle h,e,r\rangle}$;
D3 $[\, b^{(r)} \to b''^{(r)}]_{\langle h,e,r\rangle}$;
D4 $[\, b''^{(r)} \to b'''^{(r)}]_{\langle h,e,r\rangle}$;
D5 $[\, b'''^{(r)}]_{\langle h,e,r\rangle} \to b$.

- For each rule $r : [\, a]^{e_1}_{h} \to [\, b]^{e_2}_{h}[\, c]^{e_3}_{h} \in R$ of type (e), we introduce in R' the rules:

E1 $[\, a]_{\langle h,e_1\rangle} \to [\ ]_{\langle h,e_1,r\rangle} a'^{(r)}$;
E2 $a'^{(r)}[\ ]_{\langle h,e_1,r\rangle} \to [\, a^{(r)}]_{\langle h,e_1,r\rangle}$;
E3 $[\, a^{(r)}]_{\langle h,e_1,r\rangle} \to [\, b_{e_2}]_{\langle h,e_1,r\rangle}[\, c_{e_3}]_{\langle h,e_1,r\rangle}$;
E4 $[\, b_{e_2} \to b'\$_{e_2}]_{\langle h,e_1,r\rangle}$; $[\, c_{e_3} \to c'\$_{e_3}]_{\langle h,e_1,r\rangle}$;
E5 $[\, b' \to b]_{\langle h,e_1,r\rangle}$; $[\, c' \to c]_{\langle h,e_1,r\rangle}$; $[\, \$_{e_i}]_{\langle h,e_1,r\rangle} \to [\ ]_{\langle h,e_i\rangle}\$'$, for $i = 2, 3$.

- Finally, for the output of the result, we introduce in R' the rules
  $[\, a]_{s} \to [\ ]_{s} a$, for all $a \in O$,
- and then, for all labels $g \in H'$ we introduce the rule
  $[\, \# \to \#]_{g}$.

The idea is the same as in the proof of the previous lemma: instead of working with membranes with labels and polarization, $[\ ]^{e}_{h}$, we work with membranes having only labels, $[\ ]_{\langle h,e\rangle}$, with the polarizations "stored" as the second component of the labels. This time, one step of a computation in Π is simulated by five steps in Π', controlled mainly by the superscripts (r) of symbols from O', which identify the rule which is simulated. Note that r appears also in labels of the form $\langle h, e, r\rangle$, which correspond to labels of the form $\langle h', e\rangle$ in the previous proof (in the sense that these labels are always returned to labels $\langle h, e\rangle$ only in the fifth step of simulating a rule of types $(b,), (c), (d), (e)$, thus making possible the simulation of another rule).

Again, we use the trap-symbol only for ensuring the correct simulation of rules of type (b), which is different from the case of the other rules, but we do not enter here into details. With the experience of the previous proof, the reader should be able to see how the computations in Π' develop.

In all cases of rules different from type (a) it is important to note that we change the label of the membrane by sending out of it an object, one copy of which should come back in the next step. In order to ensure this, both the membrane “remembers" which kind of objects should come back, because we have r in the label, and the object “remembers" which kind of membranes has to enter, because it has the superscript (r). Because of the fact that the number of copies of objects $b'^{(r)}$ is equal to the number of membranes which send out the objects $b'^{(r)}$ and because of parallelism, each membrane which previously has sent out an object $b'^{(r)}$ will now contain an object $b^{(r)}$.

At any moment, the objects which were sent out of the system by rules of R are also sent out of μ' by the rules of R'. Consequently, the two systems Π and Π' are equivalent. $\square$

In the construction above, except for the new skin membrane, that with the label s, the membrane structure remains the same (only the labels are changed during computations).

4.2 Universality consequences

From Theorem 1 (Theorem 7.2.1 in [11]) we know that systems with rules of types $(a), (b), (c)$ are Turing complete. The proof from [11] (recalled there from [6]) uses only three membranes, arranged in two levels – hence from this point of view the premises of both lemmas from the previous section are satisfied. Unfortunately, that proof changes the polarization of the skin membrane.

A close examination of the proof shows, however, that this change is done only once, in the end of the computation. More precisely, one starts from a matrix grammar G with appearance checking in the binary normal form. The terminal matrix $(X \to \lambda, A \to x)$ of the grammar is replaced by a matrix of the form $(X \to f, A \to x)$, where f is a new symbol. The idea is that when the symbol f is introduced (actually, it is introduced as f'), the derivation G should be terminal, hence no further rule of it should be simulated in the constructed P system. To this aim, f' is sent out of the system, changing the polarization of the skin membrane from 0 to +. If the derivation in G was not terminal, then in the positively polarized skin, each nonterminal A of G evolves by a rule $A \to \#$, thus preventing the halting of the computation (we also have there the rule $\# \to \#$).

This control of the correct termination of the simulation can be achieved without changing the polarization of the skin membrane, by introducing one additional membrane, with label 4, at the same level with membranes 2 and

3, removing the rules which change the polarization of the skin or use its positive polarization, and considering the following new rules:

$$f'[\;]_4^0 \to [f']_4^+; A[\;]_4^+ \to [\#]_4^+, \text{ for all nonterminals } A \text{ of } G; [\# \to \#]_4^+.$$

The role of (the positive polarization of) the skin is played now by (the positive polarization of) membrane 4. In this way we get a system which is of type D2S0, hence we can conclude:

Theorem 5. $PsOP(a_0, b_0, c_0, e_0') = PsOP(a_0, b_0, c_0') = PsRE.$

The equalities follow from Lemmas 3, 4, from the previous change of the proof of Theorem 1, and from the observation that in the proof of Lemma 3 we use rules of type (d_0) only for simulating rules of type (d), while in the proof of Lemma 4 we use rules of types (d_0), (e_0) only in the simulation of rules of types (d), (e), respectively.

4.3 Direct universalities

Because we do not have a simulation lemma also for the case of using rules of type (b_0') for changing the labels of membranes, the universality does not follow for this case as for the other cases, and that is why we look for a direct proof of universality.

Theorem 6. $PsOP(a_0, b_0', c_0) = PsRE.$

Proof. Consider a matrix grammar $G = (N, T, S, M, F)$ with appearance checking, in the f-binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, f, \#\}$ and with the matrices of the forms introduced in Section 2. Assume that all matrices of types 2, 3, $4'$ are injectively labeled with elements of a set B.

We construct the P system of degree 2

$$\begin{aligned} \Pi &= (O, H, [\,[\;]_{X_{init}}]_1, w_1 = cA_{init}, w_{X_{init}} = \lambda, R), \\ O &= T \cup N_2 \cup \{A_m \mid A \in N_2, m \in B\} \cup \{c, c', c'', c_1, c_2, c_3, c_4, c_5, \#\}, \\ H &= N_1 \cup \{X_m \mid X \in N_1, m \in B\} \cup \{1, f\}, \end{aligned}$$

and the set R containing the following rules. We present them in blocks as used for simulating matrices of G, thus also having clear the way the system Π works.

The simulation of a matrix $m : (X \to Y, A \to x)$, with $X \in N_1, Y \in N_1 \cup \{f\}$, is done in three steps, using the next rules:

A1. $A[\;]_X \to [A_m]_{Y_m}$; $[c \to c']_1$;
A2. $[A_m]_{Y_m} \to [\;]_{Y_m} A_m$; $[c' \to c'']_1$;
A3. $[A_m \to xc]_1$; $c''[\;]_{Y_m} \to [c'']_Y$.

The first rule of the matrix is simulated by the change of the label of the inner membrane, and the correctness of this operation is obvious (one cannot simulate one rule of the matrix without simulating at the same time also the other rule).

The simulation of a matrix $m : (X \to Y, A \to \#)$, with $X, Y \in N_1$ and $A \in N_2$, is done in five steps, using the next rules:

B1. $c[\ ]_X \to [\, c_1]_{Y_m}$;
B2. $[\, c_1 \to c_2]_{Y_m}$; $A[\ ]_{Y_m} \to [\, \#]_f$;
B3. $[\, c_2]_{Y_m} \to [\ ]_{Y_m} c_3$;
B4. $[\, c_3 \to c_4 c_5]_1$;
B5. $[\, c_4 \to c]_1$; $c_5[\ ]_{Y_m} \to [\, c'']_Y$.

While the membrane with label X is used by object c, no other rule can be used. In the next step, if any copy of A is present, then it introduces the trap-object $\#$ and the computation never stops. If no A is present, then the objects c_j evolve, returning the label of the membrane to Y and recreating the auxiliary object c, for iterating the procedure.

We also consider the following rules:

$A[\ ]_f \to [\, \#]_f$, for all $A \in N_2$; $[\, \# \to \#]_f$;
$[\, a]_1 \to [\ ]_1 a$, for all $a \in T$.

The equality $\Psi_T(L(G)) = Ps(\Pi)$ easily follows from the above explanations. □

A direct proof of universality can be given also for systems using rules of the types $(a_0), (b_0), (c_0')$. We leave this task to the reader, and we give here the direct universality proof for the case of using rules of type (e_0'): only rules of types (a_0), (c_0), and (e_0') are used, thus improving the first equality from Theorem 5.

Theorem 7. $PsOP(a_0, c_0, e_0') = PsRE$.

Proof. Consider again a matrix grammar $G = (N, T, S, M, F)$ with appearance checking, in the f-binary normal form, with the notations and the assumptions from the previous proof, and construct the P system of degree 2

$$\Pi = (O, H, [\,[\ ]_{X_{init}}]_1, w_1 = \lambda, w_{X_{init}} = c_0 A_{init}, R),$$
$$O = T \cup N_2 \cup \{A_m \mid A \in N_2, m \in B\} \cup \{c, c', c_0, c_1, c_2, d, \#\},$$
$$H = N_1 \cup \{X_m \mid X \in N_1, m \in B\} \cup \{0, 1, f\},$$

and the set R containing the following rules.

The simulation of a matrix $m : (X \to Y, A \to x)$, with $X \in N_1, Y \in N_1 \cup \{f\}$, is done in three steps, using the next rules:

A1. $[\,A\,]_X \rightarrow [\,A_m\,]_{Y_m}[\,d\,]_0$;
A2. $[\,A_m \rightarrow xc\,]_{Y_m}$;
A3. $[\,c\,]_{Y_m} \rightarrow [\,c'\,]_Y[\,d\,]_0$.

Again the first rule of the matrix is simulated by the change of the label of the inner membrane (the "dummy" object d and membrane 0 play no further role).

The simulation of a matrix $m : (X \rightarrow Y, A \rightarrow \#)$, with $X, Y \in N_1$ and $A \in N_2$, is done also in three steps, using the next rules:

B1. $[\,c_0\,]_X \rightarrow [\,c_1\,]_{Y_m}[\,d\,]_0$;
B2. $[\,c_1 \rightarrow c_2\,]_{Y_m}$; $[\,A \rightarrow \#\,]_{Y_m}$;
B3. $[\,c_2\,]_{Y_m} \rightarrow [\,c_0\,]_Y[\,d\,]_0$.

While the membrane with label X is used by object c_0, no other rule can be used. In the next step, if any copy of A is present, then it introduces the trap-object $\#$ and the computation never stops. If no A is present, then the objects c_j evolve, returning the label of the membrane to Y and recreating the auxiliary object c_0, for iterating the procedure.

We also consider the following rules:

$[\,A \rightarrow \#\,]_f$, for all $A \in N_2$; $[\,\# \rightarrow \#\,]_h$, for all $h \in H$;
$[\,a\,]_f \rightarrow [\;]_f a$; $[\,a\,]_1 \rightarrow [\;]_1 a$, for all $a \in T$.

The equality $\Psi_T(L(G)) = Ps(\Pi)$ is again obvious. □

Remark 1. In the above proof, the rules of type (c_0) are only used for sending the result of a computation out of the system. Therefore, rules of types (a_0) and (e_0') are sufficient to reach universality for membrane systems with internal output.

5 Efficiency

In this section, we will show how to solve the **NP**-complete problem SAT by P systems with active membranes by using different combinations of rules of various types. The SAT problem is probably the best known **NP**-complete problem [4]; it asks whether or not for a given formula in the conjunctive normal form there is a truth-assignment of variables such that the formula assumes the value *true*.

Throughout this section we consider accepting (one also says *recognizing*) P systems with active membranes, i.e., systems which start working from an initial configuration where an encoding of a given problem is introduced as an input, and which proceed until sending out the answer, *yes* or *no*, to the problem (all computations halt). Recognizing P systems is a rigorous framework for dealing with complexity matters in membrane computing area; for more details, please refer to [12].

Here, we only briefly recall some basic notions from this area. Consider a decisional problem X. A family $\Pi_X = (\Pi_X(1), \Pi_X(2), \dots)$ of P systems (with active membranes in our case) is called *semi-uniform* (*uniform*) if its elements are constructible in polynomial time starting from $X(n)$ (from n, respectively), where $X(n)$ denotes the instance of size n of X. We say that X can be solved in polynomial (linear) time by the family Π_X if the system $\Pi_X(n)$ stops in a polynomial (linear, respectively) number of steps, sending out the object `yes` if and only if the instance $X(n)$ has a positive answer (if the system is deterministic, then there is only one computation in $\Pi_X(n)$; if the system is confluent, several computations are possible, but all of them stop and all of them send out the same object *yes* or `no` corresponding to the answer to $X(n)$).

5.1 Solving SAT *without polarizations but using label changing*

As we have noticed above, the proofs of Lemmas 3, 4 do not preserve the determinism of the simulated systems; more precisely, the constructed systems do not always halt, but any “wrong" step with respect to the starting system will lead to an endless computation. Such a behavior is not accepted in solving decidability problems with P systems, neither in the deterministic manner from [13], nor in the slightly more relaxed framework of [11], where the nondeterminism is allowed, provided that the system is confluent, and always halts.

However, as somewhat expected, P systems without polarizations, but with the possibility of changing the labels of membranes (by means of rules of types (c_0') and (e_0')) can solve **NP**-complete problems in linear time. This is illustrated below, with direct proofs, for SAT.

Before giving these proofs, it is worth noticing that rules of types $(a_0), (e_0)$ suffice in order to generate all 2^n truth-assignments for n variables from a propositional formula. Specifically, let us consider a (non-skin) membrane $[\ ]_0$ where we have the objects d_1 and a_1, and also consider the following rules:

G1 $[\, d_i \to a_{i+1}d_{i+1}]_0,\ 1 \le i < n;$
G2 $[\, a_i]_0 \to [\, t_{i,i}]_0[\, f_{i,i}]_0,\ 1 \le i \le n;$
G3 $[\, t_{i,j} \to t_{i,j+1}]_0,\quad [\, f_{i,j} \to f_{i,j+1}]_0,\ 1 \le i \le j < n.$

In each step, one “expands" one variable, starting with x_1 and ending with x_n, deterministically. The truth values $t_{i,j}, f_{i,j}$ of variables x_i have associated second subscripts j specifying the step, so that, in n steps the membrane $[\, d_1a_1]_0$ is divided in 2^n membranes, each of them containing a multiset of the form $d_nv_1v_2\dots v_n$, where $v_i \in \{t_{i,n}, f_{i,n}\}$.

In a way which will be used in the proof of the Theorem 9, by using the rules of types (a_0), (e_0) only, during the generation of truth-assignments we can also check which clauses are satisfied by the truth-assignments – we skip the details here.

Unfortunately, we do not see any way to check the truth value of the whole formula for these truth-assignments by using only rules $(a_0), (b_0), (c_0), (d_0), (e_0)$, and that is why we use below also rules for changing the labels.

Theorem 6. *P systems with rules of types $(a_0), (b_0), (c_0), (e'_0)$ can solve* $\mathtt{SAT}$*, in a confluent way, in linear time with respect to the number of variables and the number of clauses.*

Proof. Let us consider a propositional formula in the conjunctive normal form:

$$\begin{aligned}
\beta &= C_1 \wedge \ldots \wedge C_m, \\
C_i &= y_{i,1} \vee \ldots \vee y_{i,l_i},\ 1 \le i \le m, \text{ where} \\
y_{i,k} &\in \{x_j, \neg x_j \mid 1 \le j \le n\},\ 1 \le i \le m, 1 \le k \le l_i.
\end{aligned}$$

The instance β (to which the size (m,n) is associated) is encoded as a multiset over

$$V(\langle n,m \rangle) = \{x_{i,j}, x'_{i,j} \mid 1 \le i \le m, 1 \le j \le n\}.$$

The object $x_{i,j}$ represents the variable x_j appearing in the clause C_i without negation, and object $x'_{i,j}$ represents the variable x_j appearing in the clause C_i with negation. Thus, the input multiset w consists of one copy of each element from the set

$$\begin{aligned}
&\{x_{i,j} \mid x_j \in \{y_{i,k} \mid 1 \le k \le l_i\}, 1 \le i \le m, 1 \le j \le n\} \\
\cup\ &\{x'_{i,j} \mid \neg x_j \in \{y_{i,k} \mid 1 \le k \le l_i\}, 1 \le i \le m, 1 \le j \le n\}.
\end{aligned}$$

For given $(n,m) \in \mathbb{N}^2$, we construct a recognizing P system $(\Pi(\langle n,m \rangle), V(\langle n,m \rangle), 2)$ with:

$$\begin{aligned}
\Pi(\langle n,m \rangle) &= (O(\langle n,m \rangle), H, \mu, w_1, w_2, w_7, R), \\
O(\langle n,m \rangle) &= \{x_{i,j}, x'_{i,j} \mid 1 \le i \le m, 0 \le j \le n\} \\
&\cup \{d_i \mid 0 \le i \le 2n + 2m + 5\} \\
&\cup \{c_i \mid 1 \le i \le m\} \cup \{e, f_0, f_1, \mathtt{yes}, \mathtt{no}\}, \\
\mu &= [\,[\,]_2[\,]_7]_1, \\
w_1 &= \lambda,\ w_2 = w_7 = d_0, \\
H &= \{1,2,3,4,5,6,7\},
\end{aligned}$$

and the following rules (we also give explanations about the use of these rules):

Generation phase

G1 $[\,d_i]_2 \to [\,d_i]_3[\,d_i]_4, 0 \le i < n;$
G2 $[\,d_i]_l \to [\,d_{i+1}]_2[\,d_0]_1, l \in \{3,4\}, 0 \le i < n;$
G3 $[\,d_n]_2 \to [\,d_0]_5[\,d_0]_1.$

In $2n+1$ steps, 2^n membranes with label 5 are created, corresponding to the truth-assignments of the variables $x_1, \ldots, x_n$. During this process, the object d_i inside the membrane with label 3 corresponds to the *true* value of variable x_{i+1}, and the object d_i inside the membrane with label 4 corresponds to the *false* value of variable x_{i+1}. The created membranes with label 1 are dummy membranes: no rule associated with them is applied; this allows us to change the membrane labels during the computation.

G4 $[\,x_{i,j} \to x_{i,j-1}]_2,\ \ [\,x'_{i,j} \to x'_{i,j-1}]_2, 1 \le i \le m, 1 \le j \le n;$
G5 $[\,x_{i,0} \to c_i]_3,\ \ [\,x_{i,0} \to \lambda]_4, 1 \le i \le m;$
G6 $[\,x'_{i,0} \to \lambda]_3,\ \ [\,x'_{i,0} \to c_i]_4, 1 \le i \le m.$

The labels of the created membranes toggle between 2 at even steps and 3 or 4 at odd steps. Every object $x_{i,j}$ of the input evolves to $x_{i,0}$ in $2j-1$ steps. Then, it evolves to c_i in membranes where *true* value was chosen for x_j (recall that $x_{i,j} = true$ satisfies clause C_i) and is erased in membranes where *false* value was chosen for x_j. Similarly, $x'_{i,j}$ changes to c_i if $x_j = false$ and is erased if $x_j = true$. After $2n+1$ steps, the membranes with label 5 will represent all possible truth-assignments of the variables in β. Every such membrane will contain d_0 and the objects representing the clauses satisfied by the present truth-assignment.

Checking phase

C1 $[\,c_1]_5 \to [\,c_0]_6[\,d_0]_1;$
C2 $[\,c_i \to c_{i-1}]_6, 1 \le i \le m;$
C3 $[\,d_i]_6 \to [\,d_{i+1}]_5[\,d_0]_1, 0 \le i < m, [\,c_0 \to \lambda]_5;$
C4 $[\,d_m \to ef_0]_5.$

A membrane with label 5 where object c_1 appears will change the label to 6 (recall that no rule is ever applied in membranes with label 1 created by division). In a membrane with label 6, the subscripts of all objects c_j are decremented by one, and at the same time the subscript of d_i is incremented by one and the label of the membrane changes back to 5.

If in the beginning of the checking phase $c_1, \ldots, c_i$ are present ($0 \le i < m$), but c_{i+1} is absent, then the evolution of the membrane finishes after $2i$ steps with label 5, with d_i and without c_1. If all objects $c_i, 1 \le i \le m$, are present in the beginning of the checking phase, then after $2m$ steps they will all be rewritten into c_0, and d_0 will evolve into d_m (and into ef_0 in one more step).

C5 $[\,e]_5 \to [\,]_5 e;\ \ [\,f_0 \to f_1]_5;$

C6 $e[\,]_7 \rightarrow [\,e]_7$; $[\,f_1]_5 \rightarrow [\,d_{2m+2n+5}]_6[\,d_0]_1$;
C7 $e[\,]_6 \rightarrow [\,e]_6$.

If β has solutions (suppose β has s solutions, $1 \leq s \leq 2^n$), then at step $2n+2m+3$, every membrane corresponding to a solution of β ejects e into the skin region, and at the same time f_0 changes to f_1. At step $2n+2m+4$, one copy of e enters the membrane with label 7, and s membranes change label from 5 to 6 by rule $[\,f_1]_5 \rightarrow [\,e]_6[\,d_0]_1$. At step $2n+2m+5$, $s-1$ copies of e enter in $s-1$ membranes of the $s+1$ membranes with labels 6 and 7. If β has no solution, then no object e enters membrane labeled 7.

Output phase
O1 $[\,d_i \rightarrow d_{i+1}]_7$, $0 \leq i \leq 2m+2n+5$;
O2 $[\,e]_7 \rightarrow [\,\mathtt{yes}]_6[\,d_0]_1$;
O3 $[\,\mathtt{yes}]_6 \rightarrow [\,]_6\mathtt{yes}$;
O4 $[\,\mathtt{yes}]_1 \rightarrow [\,]_1\mathtt{yes}$;
O5 $[\,d_i \rightarrow \lambda]_6$, $i \in \{2n+2m+5, 2n+2m+6\}$;
O6 $[\,d_{2n+2m+6}]_7 \rightarrow [\,]_7\mathtt{no}$;
O7 $[\,\mathtt{no}]_1 \rightarrow [\,]_1\mathtt{no}$.

If β has solutions, then at step $2n+2m+4$ the membrane with label 7 receives a copy of e by rule C6. In this case, rule O2 will be applied either at step $2n+2m+5$ or at step $2n+2m+6$ (this can happen if $s > 1$ and rule C6 is applied once more at step $2n+2m+5$), changing the label of the membrane from 7 to 6. It will take two more steps to eject object yes in the skin and then into the environment. If β has no solutions, then after step $2n+2m+6$ the membrane with label 7 remains with label 7 and then rule O6 and afterwards O7 are applied, ejecting object no into the skin and then into the environment. □

If β has at least two solutions, then the behavior of this system is not deterministic: in step $2n+2m+5$ either one of the rules C6 and O2 can be applied to the membrane with label 7 (applying C6 in step $2n+2m+5$ results in one extra copy of e in membrane with label 7 and one copy of e missing in some membrane with label 6). However, the system is confluent: in either case mentioned above, after at most three further steps, the system produces the output yes and halts in the same configuration (the membrane with label 7 changes its label to 6 and the counter $d_{2n+2m+5}$ or $d_{2n+2m+6}$ is erased). From this point of view, using rules of type (c_0') allows to obtain a stronger result:

Theorem 7. *P systems with rules of types (a_0), (c_0'), (e_0) can solve* SAT*, in a deterministic way, in linear time with respect to the number of variables and the number of clauses.*

Proof. Let us consider a propositional formula in the conjunctive normal form:

$$\begin{aligned}\beta &= C_1 \wedge \ldots \wedge C_m,\\ C_i &= y_{i,1} \vee \ldots \vee y_{i,l_i},\ 1 \le i \le m, \text{ where}\\ y_{i,k} &\in \{x_j, \neg x_j \mid 1 \le j \le n\},\ 1 \le i \le m, 1 \le k \le l_i.\end{aligned}$$

The instance β is encoded as a multiset w over $\Sigma(\langle n,m\rangle)$ in the same way as in the previous proof. For given $(n,m) \in \mathbb{N}^2$, we construct a recognizing P system $(\Pi(\langle n,m\rangle), V(\langle n,m\rangle), 2)$, with

$$\begin{aligned}\Pi(\langle n,m\rangle) &= (O(\langle n,m\rangle), H, \mu, w_1, w_2, R),\\ O(\langle n,m\rangle) &= \Sigma(\langle n,m\rangle) \cup \{d_i \mid 0 \le i \le 4n+2m+4\} \cup \{e_i \mid 0 \le i < n\}\\ &\quad \cup \{c_i \mid 1 \le i \le m\} \cup \{a, t, f, u, v, \mathtt{yes}, \mathtt{no}\},\\ \mu &= [\,[\,]_2]_1,\\ w_1 &= w_2 = d_0,\\ H &= \{1,2,3,4,5,6,7\},\end{aligned}$$

and the following rules (we also explain the construction here):

Generation phase

G1 $[\, d_i \to e_i a u]_2, 0 \le i < n;$
G2 $[\, a]_2 \to [\, t]_2 [\, f]_2;$
G3 $[\, t]_2 \to [\,]_3 a, [\, f]_2 \to [\,]_4 a;$
G4 $[\, e_i \to d_{i+1}]_l, l \in \{3,4\}, 0 \le i < n;$
G5 $[\, u]_l \to [\,]_2 a, l \in \{3,4\}.$

In $4n$ steps, 2^n membranes are created, corresponding to the truth-assignments of the variables $x_1, \ldots, x_n$. During this process, object d_i inside the membrane with label 3 corresponds to the *true* value of variable x_{i+1}, and object d_i inside the membrane with label 4 corresponds to the *false* value of variable x_{i+1}. Object a is used to choose the truth-assignment of variables, and object u is used to change the membrane label back to 2.

G6 $[\, x_{i,j} \to x_{i,j-1}]_l,\ \ [\, x'_{i,j} \to x'_{i,j-1}]_l, 1 \le i \le m, 1 < j \le n, l \in \{3,4\};$
G7 $[\, x_{i,1} \to c_i]_3,\ \ [\, x_{i,1} \to \lambda]_4, 1 \le i \le m;$
G8 $[\, x'_{i,1} \to \lambda]_3,\ \ [\, x'_{i,1} \to c_i]_4, 1 \le i \le m.$

The label of the created membranes is 2 and then changes to 3 or 4 at steps $4i+3$, $0 \le i < n$. Every object $x_{i,j}$ of the input evolves to $x_{i,1}$ in $4(i-1)$ steps. Then, it evolves to c_i in membranes where *true* value was chosen for x_j (recall that $x_{i,j} = true$ satisfies clause C_i) and is erased in membranes where *false* value was chosen for x_j. Similarly, $x'_{i,j}$ changes to c_i if $x_j = false$, and is erased if $x_j = true$.

G9 $[\, d_n \to d_{n+1}v]_2$;
G10 $[\, v]_2 \to [\,]_5 a$; $[\, d_{n+1} \to d_0 u]_2$.

After step $4n + 2$, the membranes with label 5 will represent all possible truth-assignments of the variables in β. Every such membrane will contain d_0, u, and the objects representing the clauses satisfied.

Checking phase

C1 $[\, c_i \to c_{i-1}]_5$, $1 \leq i \leq m$;
C2 $[\, u]_5 \to [\,]_6 a$;
C3 $[\, c_0]_6 \to [\,]_5 a$;
C4 $[\, d_i \to d_{i+1}u]_6$, $0 \leq i < m - 1$;
C5 $[\, d_{m-1} \to d_m]_6$.

By expelling object u, the membrane changes label from 5 to 6. At the same time the subscripts of all objects c_j are decremented by one. A membrane with label 6 where object c_0 appears will change the label back to 5. At the same time the subscript of d_i is incremented by one and u is reproduced (except for $i = m - 1$).

If in the beginning of the checking phase $c_1, \ldots, c_i$ are present ($1 \leq i < m$), but c_{i+1} is absent, then after $2i + 1$ steps rule C3 will no longer be applicable and the membrane will have label 6, no object c_0, and will never change the label again. After $m + i + 1$ steps from the beginning of the checking phase the membrane will stop evolving. If all objects c_i, $1 \leq i \leq m$, are present in the beginning of the checking phase, then after $2m$ steps they all will have been erased, d_0 will have evolved into d_m and the membrane label will be 5.

Output phase

O1 $[\, d_m]_5 \to [\,]_5 \texttt{yes}$;
O2 $[\, \texttt{yes}]_1 \to [\,]_7 \texttt{yes}$;
O3 $[\, d_i \to d_{i+1}]_1$, $0 \leq i \leq 4n + 2m + 3$;
O4 $[\, d_{4n+2m+4}]_1 \to [\,]_1 \texttt{no}$.

At step $4n + 2m + 3$, every membrane corresponding to a solution of β expels yes in the skin region, and in the next step one copy of yes (if any) is ejected into the environment, changing the label of the skin from 1 to 7. If β has no solutions, then after step $4n + 2m + 4$ the skin membrane remains with label 1 and then rule O4 is applied, ejecting the object no into the environment. □

5.2 Parallel communication

In P systems with active membranes, the evolution rules (those of type (a)) are typically considered as only using objects, while the communication, dissolution and division membranes as using both objects and membranes, and hence cannot be applied in parallel, because a *conflict* could appear as a result of a simultaneous application of rules changing membrane polarizations (or labels) in a different way.

However, if we forbid the communication operations to change labels (type (b_0) or (c_0)), then we could regard them as not using membranes and apply them in parallel, like the evolution rules, as is done, for instance, in the symport/antiport P systems in [8] and in P systems with boundary rules in [2]. We use subscript p to denote the property of parallel application of rules of the following forms.

(b_{0p}) $a[\,]_h \to [\,b]_h$, where $a, b \in O$ and $h \in H$;
(c_{0p}) $[\,a]_h \to [\,]_h b$, where $a, b \in O$ and $h \in H$.

P systems with membrane division with changing labels, and with parallel application of both evolution rules and communication rules of type (b_0) turn out to be able to solve SAT in linear time in a deterministic way, thus improving from this point of view the result in Theorem 6. The universality of systems with parallel communication remains as an open question.

Theorem 8. *P systems with rules of types $(a_0), (b_{0p}), (c_0), (e_0')$ can solve* SAT*, in a deterministic way, in linear time with respect to the number of variables and the number of clauses.*

Proof. Following the generation phase and rules C1–C3 in Theorem 6, we replace the remaining part of the construction with:

C4 $[\,d_m]_5 \to [\,]_5 e$;
C5 $e[\,]_7 \to [\,e]_7$.

At step $2n + 2m + 2$, every membrane corresponding to a solution of β ejects e in the skin. At step $2n + 2m + 3$, all objects e move in parallel into a membrane with label 7.

C6 $[\,e]_7 \to [\,\mathtt{yes}]_1 [\,d_0]_1$;
C7 $[\,\mathtt{yes}]_1 \to [\,]_1 \mathtt{yes}$;
C8 $[\,d_i \to d_{i+1}]_7$, $0 \le i \le 2n + 2m + 3$;
C9 $[\,d_{2n+2m+4}]_7 \to [\,]_7 \mathtt{no}$, $[\,\mathtt{no}]_1 \to [\,]_1 \mathtt{no}$.

If β has a solution, then we replace one copy of e with yes, changing the label from 7 to 1, send yes out of that membrane and then eject it in the environment. Otherwise, after step $2n + 2m + 4$ the membrane with label 7 will not change its label, so no will be sent out of it and then ejected in the environment. □

Remark 2. In Theorem 7, no rules of type (b_0) were used, so its statement remains valid also for parallel communication “in" (rules of type (b_{0p})).

5.3 Solving SAT *without polarizations and without changing labels*

In the brute-force algorithms as those in Section 5, the first phase produces all 2^n truth-assignments for the n variables used and the list of clauses satisfied. As we have noticed at the beginning of Section 5, this can be done without using polarization and without using the label changing possibilities. Actually, rules of types (a_0) and (e_0) suffice.

The second phase is to check whether there exists a membrane containing a given set of symbols. If the second phase started from a special membrane structure (of a form we will see below: with each truth-assignment separately enclosed in m membranes embedded in each other, corresponding to the m clauses of a formula – see Fig. 1 for a pictorial representation), then one could solve the problem without polarizations and without changing labels, moreover, only using rules of types (a_0), (c_0), and (d_0). So, the problem remains to produce the membrane structure of this “special" form – and this can be achieved by using non-elementary membrane division rules without polarization and without changing the labels of membranes.

The rules we are using will be of the form

(f_0) $[\,[\,]_i[\,]_j]_k \to [\,[\,]_i]_k[\,[\,]_j]_k$, where i, j, k are labels.

The meaning of such a rule is that if two membranes with labels i, j are placed inside a membrane with label k, then the membrane k is divided so that one of the new membranes k contains membrane i and the other one contains membrane j; all membranes and objects placed inside membranes i and j, as well as all membranes and objects from membrane k placed outside membranes i and j, are reproduced in the new copies of membrane k. As usual, the membranes different from i, j, k (those not involved in this rule) evolve in the standard non-deterministic maximally parallel manner.

Rules for dividing non-elementary membranes are already known to be very powerful – illustration can be found in, e.g., [1,16]. As we will see immediately, they are powerful even in the restricted case where no polarization is used and the labels of membranes are not changed.

Theorem 9. *P systems with rules of types* $(a_0), (c_0), (d_0), (e_0), (f_0)$*, constructed in a semi-uniform manner, can solve* SAT*, in a deterministic way, in linear time with respect to the number of variables and the number of clauses.*

Proof. Let us consider a propositional formula in the conjunctive normal form:

$$\begin{aligned}
&\beta = C_1 \wedge \ldots \wedge C_m,\\
&C_i = y_{i,1} \vee \ldots \vee y_{i,l_i},\ 1 \leq i \leq m, \text{ where}\\
&y_{i,k} \in \{x_j, \neg x_j \mid 1 \leq j \leq n\},\ 1 \leq i \leq m, 1 \leq k \leq l_i.
\end{aligned}$$

The instance β of SAT will be encoded in the rules of the P system by multisets v_j and v'_j of symbols, corresponding to the clauses satisfied by *true* and *false* assignment of x_j, respectively:

$$v_j = \{c_i \mid x_j \in \{y_{i,k} \mid 1 \le k \le l_i\}, 1 \le i \le m\}, 1 \le j \le n,$$
$$v'_j = \{c_i \mid \neg x_j \in \{y_{i,k} \mid 1 \le k \le l_i\}, 1 \le i \le m\}, 1 \le j \le n.$$

We construct the P system

$$\begin{aligned} \Pi &= (O, H, \mu, w_0, \dots, w_{m+3}, R), \text{ with} \\ O &= \{d_i \mid 0 \le i \le 2n+2m+2\} \cup \{a_i, t_i, f_i \mid 1 \le i \le n\} \\ &\quad \cup \{c_i \mid 0 \le i \le m\} \cup \{\texttt{yes}, \texttt{no}\}, \\ \mu &= [\,[\,\cdots[\,[\,]_0]_1 \cdots]_{m+2}]_{m+3}, \\ w_0 &= w_{m+1} = d_0, \\ w_i &= \lambda, i \notin \{0, m+1\}, \\ H &= \{0, \dots, m+3\}, \end{aligned}$$

and the following rules (we accompany them with explanations about their use):

Generation phase

G1 $[\,d_{2i} \to a_{i+1}d_{2i+1}]_0$, for all $0 \le i < n$; and
$[\,d_{2i-1} \to d_{2i}]_0, 1 \le i \le n$; $\quad [\,d_{2n+i} \to d_{2n+i+1}]_0, 0 \le i < m.$

We count to $2n + m$, which is the time needed for producing all 2^n truth-assignments for the n variables, as well as membrane sub-structures which will examine the truth value of formula β for each of these truth-assignments; this counting is done in the central membrane; moreover during the first n odd steps, symbols $a_1, \dots, a_n$ are subsequently produced.

G2 $[\,a_i]_0 \to [\,t_i]_0[\,f_i]_0, 1 \le i \le n.$

In membrane 0, we subsequently choose each variable x_i, $1 \le i \le n$, and both values $true$ and $false$ are associated with it, in form of objects t_i and f_i, which are separated in two membranes with label 0. The division of membrane 0 is triggered by the objects a_i, which are introduced by the first rule from group G1 in odd steps; this is important in interleaving the use of these rules (hence the division of membrane 0) with the use of the rules of group G4, for dividing membranes placed above membrane 0.

G3 $[\,t_i \to v_i]_0, \quad [\,f_i \to v'_i]_0, 0 \le i < n.$

In membrane 0, we subsequently look for the clauses satisfied by the truth-assignments of each variable x_i, $1 \le i \le n$. After $2n+m$ steps, if there is at least one membrane with label 0 which contains all the symbols $c_1, \dots, c_m$, this means that the truth-assignment from that membrane satisfies all clauses, hence it satisfies formula β. Otherwise (if in no membrane with label 0 we get all objects $c_1, c_2, \dots, c_m$), the formula β is not satisfiable.

G4 $[\,[\,]_i[\,]_i]_{i+1} \to [\,[\,]_i]_{i+1}[\,[\,]_i]_{i+1}$, $0 \leq i < m$.

These are division rules for membranes with label $0, 1, \ldots, m$, to be used for the central membrane 0 in steps which alternate with the use of the first rule of type G1. The division of a membrane with label 1 is then propagated from lower levels to upper levels of the membrane structure and the membranes are continuously divided until also a membrane with label m has been divided. In the following cycle of the division process, the same holds, resulting in the structure as shown in Fig. 1 after $2n + m$ steps.

G5 $[\, d_{2n+m}]_0 \to c_0$.

After $2n + m$ steps, each copy of membrane with label 0 is dissolved and the contents is released into the surrounding membrane, which is labeled with 1.

Checking phase

C1 $[\, c_i]_i \to c_i$, $1 \leq i \leq m$.

A membrane with label j, $1 \leq j \leq m$, is dissolved if and only if c_j appears in it (i.e., clause C_j is satisfied by the current truth-assignment); if this is the case, then the truth-assignment associated with the membrane is released in the surrounding membrane. Otherwise, the truth-assignment remains blocked in membrane j and never used at the next steps by the membranes placed above.

C2 $[\, c_0]_{m+1} \to c_0$.

The fact the object c_0 appears in the membrane with the label $m + 1$ means that there is a truth-assignment which satisfies the formula β. In this case, the membrane with label $m + 1$ is dissolved and the contents are released into the membrane with label $m + 2$. Otherwise, the formula is not satisfiable, and the membrane with label $m + 1$ will not dissolve.

C3 $[\, d_i \to d_{i+1}]_{m+1}$, $0 \leq i \leq 2n + 2m + 1$.

At the same time as the membrane with label $m + 1$ is dissolved (at step $2n+2m+1$), the object $d_{2n+2m+1}$ evolves to $d_{2n+2m+2}$, and then is released to the membrane with label $m + 2$.

Output phase

O1 $[\, d_{2n+2m+2}]_{m+2} \to \texttt{yes}$.
O2 $[\, a]_{m+3} \to [\,]_{m+3}a$, $a \in \{\texttt{yes}, \texttt{no}\}$.

In the next two steps, the object yes is produced, and then sent out to the environment.

O3 $[\, d_{2n+2m+2}]_{m+1} \to \texttt{no}$.
O4 $[\texttt{no}]_{m+2} \to \texttt{no}$.

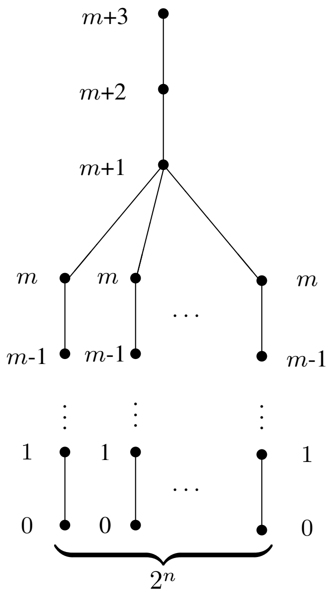

Fig. 1. The membrane structure of the system Π after $2n + m$ steps

If the formula is not satisfiable, then the object $d_{2n+2m+1}$ remains in the membrane with label $m + 1$, which produces the object no, ejecting it into the membrane with label $m + 2$, then into the membrane with label $m + 3$, and finally into the environment.

Therefore, in $2n + 2m + 3$ steps the system halts and sends into the environment one of the objects `yes`, `no`, indicating whether or not the formula β is satisfiable.

It is easy to see that the system Π can be constructed in a polynomial time starting from β and this concludes the proof. □

Remark 3. Rules of type (c_0) are only needed to output the result in the environment, so this type can be omitted if we consider internal output: exactly one of objects `yes` and `no` will be introduced in the skin membrane in the last step of the computation.

6 Final remarks

With the goal of removing the polarizations from P systems with active membranes, we have investigated the possibility to allow instead to change the labels of membranes, and we were successful in the case of rules for sending objects out of a membrane (of type (c)) and in the case of rules for dividing membranes (of type (e)) – losing however the determinism. The case of using rules of type (b) (introducing objects into membranes) for changing the labels has remained open in what concerns the simulation results – as well as the possibility to solve SAT in polynomial time – but not in what concerns the universality.

The use of non-polarized membranes suggests further possibilities in what concerns the application of rules. For instance, as already mentioned in Section 5.2, one of the reasons to use the rules of types $(b), (c)$ in a sequential way was the polarization change (using several rules at the same time could lead to polarization conflicts). The same reason prevents using rules of types $(b_0'), (c_0')$ in a parallel manner. When no polarizations are present and no label is changed, these difficulties do not appear, hence we can use also rules of types $(b_0), (c_0)$ in parallel: all objects which can enter or exit a membrane have to do it at the same time, in the maximally parallel manner.

On the other hand, we can allow also rules of type (a) to change the polarization or the label of the membrane – and then such a rule should be applied in a sequential manner, not to lead to label conflicts. We write such a rule in the form $[\, a]_h^{e_1} \rightarrow [\, v]_h^{e_2}$, or $[\, a]_{h_1} \rightarrow [\, v]_{h_2}$.

In total, we get three criteria to classify the rules: changing or not polarizations, changing or not labels of membranes, using the rules in parallel or sequentially. On the other hand, we have rules of five forms (six, if we also consider rules for dividing non-elementary membranes), each one being of several possible types with respect to the previous classification. A lot of classes of P systems are obtained by combining these possibilities, a small “jungle" which is worth exploring, looking for results of three types: simulation lemmas among different classes of P systems, universality results (as a consequence of possible simulation lemmas or directly proven), efficiency results. We hope to return to this topic in a forthcoming paper.

Acknowledgements. Thanks are due to two anonymous referees for their comments and suggestions improving the presentation of this paper. The first author is supported by the project TC2002-04220-C03-02 of the Research Group on Mathematical Linguistics, Tarragona. The first author also acknowledges the Moldovan Research and Development Association (MRDA) and the U.S. Civilian Research and Development Foundation (CRDF), Award No. MM2-3034. The second author is supported by the grant DGU-SB2001-0092 from Spanish Ministry for Education, Culture, and Sport, National Natural Science Foundation of China (Grant No. 60373089), and Huazhong University of Science and Technology Foundation.

References

1. Alhazov, A., Martín-Vide, M., Pan, L. (2003) Solving a **PSPACE**-complete problem by P systems with restricted active membranes. Fundamenta Informaticae 58(2): 66–77
2. Bernardini, F., Manca, V. (2003) P systems with boundary rules. In: Păun, Gh., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane computing. Proc. WMC Curtea de Argeş, 2002, LNCS 2597. Springer, Berlin Heidelberg New York, pp 107–118
3. Dassow, J., Păun, Gh. (1989) Regulated rewriting in formal language theory. Springer, Berlin Heidelberg New York

4. Garey, M.R., Johnson, D.J. (1979) Computers and intractability. A guide to the theory of np-completeness. W.H. Freeman, San Francisco
5. Hauschild, D., Jantzen, M. (1994) Petri nets algorithms in the theory of matrix grammars. Acta Informatica 31: 719–728
6. Madhu, M., Krithivasan, K. (2002) Improved results about the universality of P systems. Bulletin of the EATCS 76: 162–168
7. Papadimitriou, Ch.P. (1994) Computational complexity. Addison-Wesley, Reading, MA
8. Păun, A., Păun, Gh. (2002) The power of communication: P systems with symport/antiport. New Generation Computers 20(3): 295–306
9. Păun, Gh. (2000) Computing with membranes. J. Computer System Sciences 61(1): 108–143
10. Păun, Gh. (2001) P systems with active membranes: attacking **NP**-complete problems. J. Automata, Languages Combinatorics 6(1): 75–90
11. Păun, Gh. (2002) Computing with membranes: an introduction. Springer, Berlin Heidelberg New York
12. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F. (2003) Complexity classes in models of cellular computation with membranes. Natural Computing 2(3): 265–285
13. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F. (2002) Teoría de la Complejidad en Modelos de Computatión Celular con Membranas. Editorial Kronos, Sevilla
14. Rozenberg, G., Salomaa, A. (eds.) (1997) Handbook of formal languages (3 vols). Springer, Berlin Heidelberg New York
15. Salomaa, A. (1973) Formal languages. Academic Press, New York
16. Sosík, P. (2003) Solving a **PSPACE**-complete problem by P systems with active membranes. In: Cavaliere, M., Martín-Vide, C., Păun, Gh. (eds.) Proc. of the Brainstorming Week on Membrane Computing, Report GRLMC 26/03, pp. 305–312
17. Zandron, C., Mauri, G., Ferretti, C. (2000) Universality and normal forms on membrane systems. In: Freund, R., Kelemenova, A. (eds.) Proc. Int. Workshop on Grammar Systems 2000, Bad Ischl, Austria, pp. 61–74