P Systems with Tables of Rules

Gheorghe PĂUN[1,2], Mario PÉREZ-JIMÉNEZ[2], Agustín RISCOS-NÚÑEZ[2]

[1]Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania

[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Technical Higher School of Computer Science Engineering
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: {gpaun, marper, ariscosn}@us.es

Abstract. In the last time, several efforts were made in order to remove the polarization of membranes from P systems with active membranes; the present paper is a contribution in this respect. In order to compensate the loss of power represented by avoiding polarizations, we introduce *tables* of rules: each membrane has associated several sets of rules, one of which is non-deterministically chosen in each computation step. Three universality results for tabled P systems are given, trying to use rules of as few as possible types. Then, we consider tables with *obligatory rules* – rules which must be applied at least once when the table is applied. Systems which use tables with at most one obligatory rule are proven to be able to solve SAT problem in linear time. Several open problems are also formulated.

1 Introduction

In membrane computing, the P systems with active membranes have a special place, because of the fact that they provide biologically inspired means to solve computationally hard problems: by using the possibility to divide membranes, one can create an exponential working space in a linear time, which can then be used in a parallel computation for solving, e.g., **NP**-complete problems in polynomial or even linear time. Details can be found in [7], [8], as well as in the comprehensive page from the web address http://psystems.disco.unimib.it).

One of the important ingredients of P systems with active membranes is the polarization of membranes: besides a label, each membrane also has an "electrical charge", one of + (positive), − (negative), 0 (neutral). These electrical charges correspond only remotely to biological facts; by sending ions outside, cells and cell compartments can get polarizations, but this is not a very common phenomenon. Starting from this observation and also as a mathematical challenge, in the last time several efforts were made to avoid using polarizations.

However, the question seems not to be a simple one, and the best result obtained so far was to reduce to two the number of "electrical charges"; this is achieved in [1], where both the universality and the possibility of solving SAT in linear time are proven for P systems with active membranes and only two polarizations. When completely removing the polarizations, similar results are obtained (see [2], [3]) only by compensating the loss of power (of "programming" possibilities) by using additional ingredients, such as the possibility of changing the labels of membranes, division of non-elementary membranes, etc.

The present paper goes into the same direction of research: we get rid of polarizations and we "pay" this by structuring the sets of rules associated with each membrane by considering *tables* of rules, like in Lindenmayer systems. Specifically, several sets of rules are associated with each membrane, and in each step of a computation we nondeterministically choose one of these sets, its rules are used in the maximally parallel manner. The use of tables can have a biological motivation, in the same way as the tables from L systems theory have a biological origin: the change of environmental conditions (for instance, of seasons) can select specific evolution rules for different times (different seasons).

The use of tables proves to be helpful in what concerns the computing power: we get universality for systems of a rather reduced forms, with only a few types of rules used, and without polarizations.

An important *problem* remains unsolved: can tables compensate polarizations also in what concerns the possibility to solve hard problems in polynomial time? A possible negative answer to this problem would be a very nice finding: in view of the result from [1], it would follow that passing from one polarization (all membranes neutral) to two polarizations makes possible the step from the complexity class **P** to **NP**.

If, however, we add a further ingredient – at the first sight not very powerful – to tabled P systems, namely designating in each table rules which should be used at least once when applying the table, then we can solve SAT in linear time. The construction uses at most one obligatory rule in each table.

2 P Systems with Active Membranes

We assume the reader to be familiar with basic elements of membrane computing, but, for the sake of completeness, we recall here the definition of the class of P systems we work with, those with active membranes (and electrical charges).

Such a system is a construct

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

where:

1. $m \geq 1$ (the initial degree of the system);
2. O is the alphabet of *objects*;
3. μ is a *membrane structure*, consisting of m membranes, labeled in a one-to-one manner with elements of $H = \{1, 2, \ldots, m\}$;
4. $w_1, \ldots, w_m$ are strings over O, describing the *multisets of objects* placed in the m regions of μ;
5. 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).

We have omitted the rules for dividing non-elementary membranes, usually identified as being "of type (f)".

In the customary definition of P systems with active membranes, the initial membranes of μ are not necessarily labeled in a one-to-one manner, but there is no loss of generality in the assumption that the labels are unique: we can relabel the membranes with the same label and then duplicate the necessary rules. Moreover, because in what follows we only consider that by membrane division we obtain membranes with the same label, the labels present in the system are always from the set $\{1, 2, \ldots, m\}$ present at the beginning (maybe some of them used several times, because of the division of membranes). Therefore, the set H of labels is specified by μ, it can be omitted when specifying the system.

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 number of objects expelled into the environment during the computation; the set of numbers computed in this way by the various halting computations in Π is denoted by $N(\Pi)$.

By $NOP_{m,n,p}(pol_3, a, b, c, d, e)$ we denote the family of sets $N(\Pi)$ computed as sketched above by systems starting with at most m membranes, using membranes of at most n types,

at most p membranes being simultaneously present, and using all types of rules; when rules of a certain type are not used the corresponding letter a, b, c, d, e will be missing. Also, when membrane division rules are not used, we will specify only the number of membranes in the initial configuration (hence, only m) as a subscript of NOP. The parameter pol_3 indicates the fact that one uses three polarizations.

Further details can be found in [7] – including the proof of the following result. (We denote by REG, CF, CS, RE the families of regular, context-free, context-sensitive, and of recursively enumerable languages. In general, for a family FL of languages, NFL denotes the family of length sets of languages in FL. Therefore, NRE is the family of Turing computable sets of natural numbers.)

Theorem 2.1 $NOP_3(pol_3, a, b, c) = NRE$.

The number of polarizations were decreased to two in [1]; with the previous notations, the result can be written as:

Theorem 2.2 $NOP_2(pol_2, a, c) = NRE$.

Note that the result from Theorem 2.1 was improved both in the number of polarizations and the number of membranes, while the used rules are of the same types.

In [3] and [2] rules of types $(a) - (e)$ without polarizations were considered. Because "no polarization" means "neutral polarization", we add the subscript 0 to the previous letters identifying the five types $(a_0) - (e_0)$ of rules.

The power of polarizationless P systems with active membranes is not precisely known, but it was shown in [2] that they are able to compute at least the Parikh images of languages generated by matrix grammars without appearance checking.

Because the notion of a matrix grammar will be also used below, we introduce it here in its general form.

A *matrix grammar* (with appearance checking) is a construct $G = (N, T, S, M, F)$, where N and T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and F is a set of occurrences of rules in M (N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either $w_i = w'_i A_i w''_i, w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F. (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied – therefore we say that these rules are applied in the *appearance checking* mode.)

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.

It is known that $CF \subset MAT \subset MAT_{ac} = RE$, $NREG = NCF = NMAT \subset NCS$, (for instance, the one-letter languages in MAT are known to be regular, [6]).

A matrix grammar $G = (N, T, S, M, F)$ is said to be 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 $(S \to X_0A_0)$, 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 each matrix grammar there is an equivalent matrix grammar in the binary normal form. Details can be found in [4].

3 Tables of Rules

In the "standard" P systems with active membranes there is specified only one set of rules; because the membranes are present in the rules, we precisely know where each rule is to be applied with respect to each membrane. A possible generalization is to consider several sets of rules – for uniformity with L systems, we call them *tables* – such that in each step of a computation a table is used, non-deterministically chosen (the rules of the selected table are applied in the maximally parallel manner, as mentioned in the previous section).

This case corresponds to having *global tables*; a more relaxed variant is to consider *local tables*, sets of rules associated with each membrane.

Specifically, for each membrane i we can consider sets $R_{i,1}, \ldots, R_{i,k_i}$ of rules, for some $k_i \geq 1$, all of the rules involving membrane i. In a step of a computation, we apply the rules from one of the tables associated with each membrane, as usual, in the maximally parallel non-deterministic manner with respect to the chosen table.

If we are allowed to "evolve" a region by means of a table for which no rule is actually applied, then the local tables can be combined in global tables, hence in this case the local version is weaker than the global one. However, there is no difference from the computational point of view (at least in the cases investigated in the next section): systems with local tables (and restricted types of rules) are equivalent with Turing machines; moreover, the proofs are based on systems with one or two membranes, with the "main work" of two-membranes systems done in the inner membrane, hence choosing tables which change nothing in one of the regions do not change the generated set of numbers.

In what follows we will consider only local tables, that is why we choose a more restricted – also, more natural – definition of a transition step: if there are tables by which a region can effectively evolve (at least a rule of these tables can be effectively applied), then one of these tables must be chosen. Otherwise stated, we cannot choose a table with no applicable rule if there are tables with applicable rules. This restriction both corresponds to the notions of parallelism and synchronization, basic in membrane computing, and it is also useful in the proofs below.

In systems with tables (either local or global) we have two levels of non-determinism: in each step we first non-deterministically choose one table (in the local case, associated with each membrane), and then we use the rules of the chosen table in a non-deterministic manner (observing the restriction of maximal parallelism for the chosen table). The standard definition of P systems corresponds to the case where we have only one table (at the level of the system).

The fact that we use (local) tables is indicated by adding *tab* to the notations from the previous section.

We do not know whether the number of tables associated with membranes matters (that is, whether it induces an infinite hierarchy of the computed sets of numbers) or normal form theorems like that known for ET0L systems (two tables are enough, see [9]) are true also in our case. In view of this *open problem* it could be better to indicate also the maximal number of tables used, writing tab_s for using at most s tables, but we do not deal with this aspect here.

The usefulness of using tables is intuitively obvious, because by clustering the rules in "teams of rules" we can control in a more careful way the work of the system. This is illustrated also by the following simple **example**: consider the system

$$\begin{aligned} \Pi &= (\{a,b\}, [\ \ ]_1, a, R_{1,1}, R_{1,2}, R_{1,3}), \\ R_{1,1} &= \{[\ a \to aa\ ]_1\}, \\ R_{1,2} &= \{[\ a \to b\ ]_1\}, \\ R_{1,3} &= \{[\ b\ ]_1 \to a\}. \end{aligned}$$

After using $n \geq 0$ times the first table (thus producing 2^n copies of a), we can end the computation by using once the second table, and then 2^n times the third one. Consequently, $N(\Pi) = \{2^n \mid n \geq 1\} \in NOP_1(tab, a_0, c_0)$, a set of numbers which is not in $NMAT$.

4 Universality Results

The usefulness of tables is illustrated also by the results below: the computational universality is obtained without polarizations for various reduced combinations of types of rules.

The first result uses rules of the first three types (hence not membrane dissolution or membrane division operations).

Theorem 4.1 $NOP_2(tab, a_0, b_0, c_0) = NRE$.

Proof. Let us consider a matrix grammar with appearance checking $G = (N, \{a\}, S, M, F)$ in the binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, \#\}$ and with matrices of the four types mentioned in Section 2. All matrices of M are supposed to be labeled in an injective manner with $m_i, 1 \leq i \leq n$ (hence i uniquely identifies the matrix). Each terminal matrix $(X \to \lambda, A \to x)$ is replaced with $(X \to f, A \to x)$, where f is a new symbol (the label of the matrix remains unchanged).

We construct the tabled P system with active membranes, Π, with the components:

$$\begin{aligned} O &= N_1 \cup N_2 \cup \{Z_i, Z_i', \langle i \rangle \mid 1 \leq i \leq n\} \cup \{a, a', e, f, \#\}, \\ \mu &= [\ [\ \ ]_2\ ]_1, \\ w_1 &= \lambda, \\ w_2 &= X_0 A_0 e, \text{ where } (S \to X_0 A_0) \text{ is the initial matrix of } G, \end{aligned}$$

and the following tables (by U we denote the set $N_1 \cup \{Z_i, Z_i', \langle i \rangle \mid 1 \leq i \leq n\}$).

1. For each matrix $m_i : (X \to Y, A \to x)$ in M of types 2 or 4, we consider the tables

$$\begin{aligned} R_{2,i} &= \{[\ X \to Z_i\ ]_2,\ [\ A\ ]_2 \to [\ \ ]_2\langle i\rangle,\ [\ e\ ]_2 \to \#\} \\ &\cup \{[\ \alpha \to \#\ ]_2 \mid \alpha \in U\}, \\ R'_{2,i} &= \{[\ Z_i \to Z'_i\ ]_2,\ \langle i\rangle[\ \ ]_2 \to [\ \langle i\rangle\ ]_2\} \\ &\cup \{[\ \alpha \to \#\ ]_2 \mid \alpha \in U\}, \\ R''_{2,i} &= \{[\ Z'_i \to \lambda\ ]_2,\ [\ \langle i\rangle \to xY\ ]_2\} \\ &\cup \{[\ \alpha \to \#\ ]_2 \mid \alpha \in U\}. \end{aligned}$$

2. For each matrix $m_i : (X \to Y, A \to \#)$ in M of type 3, we consider the table

$$\begin{aligned} R_{2,i} &= \{[\ X \to Y\ ]_2,\ [\ A \to \#\ ]_2\} \\ &\cup \{[\ \alpha \to \#\ ]_2 \mid \alpha \in U\}. \end{aligned}$$

3. We also consider the following tables:

$$\begin{aligned} R_{2,f} &= \{[\ f \to \lambda\ ]_2\} \\ &\cup \{[\ \alpha \to \#\ ]_2 \mid \alpha \in U \cup N_2\}, \\ R_{2,a} &= \{[\ a\ ]_2 \to [\ \ ]_2 a',\ [\ \# \to \#\ ]_2\}, \\ R_1 &= \{[\ a'\ ]_1 \to [\ \ ]_1 a,\ [\ \# \to \#\ ]_1\}. \end{aligned}$$

We have the equality $N(\Pi) = \{n \mid a^n \in L(G)\}$. Indeed, we start with the multiset X_0A_0e in the central membrane; assume that we have here a multiset Xwe for some $X \in N_1$ and $w \in (N_2 \cup \{a\})^*$. There is only one table for membrane 1, sending out a copy of a (provided that there are copies of a' in the skin region), and using the trap-rule $\# \to \#$ provided that the object $\#$ is present; in this latter case, the computation will never stop. If applied in membrane 2 when Xwe is here, the table $R_{2,f}$ will introduce the trap-object $\#$, and this happens also if we use any table of the forms $R'_{2,i}, R''_{2,i}$. Thus, we can apply only a table of type $R_{2,i}$ for m_i a matrix of M. That matrix should be either of the form $m_i : (X \to Y, A \to x)$ (of type 2 or of type 4), or of the form $m_i : (X \to Y, A \to \#)$ (of type 3): if the first rule of the matrix is $\alpha \to \beta$ with $\alpha \neq X$, then the trap-object is introduced.

The case of a matrix of type 3 is simpler: if A is present, then the trap-object is introduced, and the computation will never stop (because of the table $R_{2,a}$, which can be used forever). If A is not present, then we just change X into Y. Thus, the simulation of the matrix m_i of type 3 is correct.

If we choose to simulate a matrix of types 2 or 4, then it must have the second rule of the form $A \to x$, for A as specified by the table $R_{2,i}$: if the rule $[\ A\ ]_2 \to [\ \ ]_2\langle i\rangle$ is not used, thus "keeping busy" the membrane, then the rule $[\ e\ ]_2 \to \#$ must be used, and the computation will never stop (table R_1 can be applied forever).

In the next step we have to continue the simulation of the matrix m_i by using the corresponding table $R'_{2,i}$. This is the only table which will not introduce $\#$ which can be applied without introducing the trap-object. In this way, $\langle i\rangle$ comes back to membrane 2, and Z_i is replaced by Z'_i. In the next step, again only one table can be used without introducing the trap-object, namely $R''_{2,i}$. It erases the object Z'_i and replaces $\langle i\rangle$ with xY, thus completing the simulation of the matrix.

At any moment, if any object a is present in membrane 2, then table $R_{2,a}$ can be used and a is sent out (first transformed into a' in the skin region).

The system is returned to a configuration with the contents of membrane 2 as in the beginning, hence the process can be iterated. When the object f is introduced, no table $R_{2,i}, R'_{2,i}, R''_{2,i}$ can be used. By means of $R_{2,f}$ we check whether any symbol from N_2 is present, hence whether the derivation in G is terminal. The computation in Π ends by sending out all copies of a, hence $N(\Pi)$ equals the length set of the language $L(G)$. □

In the previous proof, the role of rules of type $(b_0), (c_0)$ (besides sending the result outside the system) was to ensure that only one object A is replaced by x, thus correctly simulating the second rule of a matrix $(X \to Y, A \to x)$ of types 2 or 4. This can be done also by using rules of type (e_0).

Theorem 4.2 $NOP_{2,2,3}(tab, a_0, c_0, e_0) = NRE$.

Proof. As above, we consider a matrix grammar with appearance checking $G = (N, \{a\}, S, M, F)$ in the binary normal form, with the matrices of M labeled in an injective manner with $m_i, 1 \leq i \leq n$, and each terminal matrix $(X \to \lambda, A \to x)$ replaced with $(X \to f, A \to x)$, where f is a new symbol.

We now construct the tabled P system with active membranes Π, with the components:

$$\begin{aligned} O &= N_1 \cup N_2 \cup \{Z_i, \langle i \rangle \mid 1 \leq i \leq n\} \cup \{a, a', d, e, f, \#\}, \\ \mu &= [\,[\quad]_2\,]_1, \\ w_1 &= \lambda, \\ w_2 &= X_0 A_0 e, \text{ where } (S \to X_0 A_0) \text{ is the initial matrix of } G, \end{aligned}$$

and the following tables (by U we denote the set $N_1 \cup \{Z_i, \langle i \rangle \mid 1 \leq i \leq n\}$).

1. For each matrix $m_i : (X \to Y, A \to x)$ in M of types 2 or 4, we consider the tables

$$\begin{aligned} R_{2,i} &= \{[\, X \to Z_i\,]_2,\ [\, A\,]_2 \to [\, \langle i \rangle\,]_2 [\, d\,]_2, \\ &\quad\ [\, d \to \#\,]_2,\ [\, e\,]_2 \to [\, \#\,]_2 [\, \#\,]_2\} \\ &\cup \{[\, \alpha \to \#\,]_2 \mid \alpha \in U \cup \{a\}\}, \\ R'_{2,i} &= \{[\, Z_i \to \lambda\,]_2,\ [\, \langle i \rangle \to xY\,]_2,\ [\, d \to \#\,]_2\} \\ &\cup \{[\, \alpha \to \#\,]_2 \mid \alpha \in U \cup \{a\}\}. \end{aligned}$$

2. For each matrix $m_i : (X \to Y, A \to \#)$ in M of type 3, we consider the table

$$\begin{aligned} R_{2,i} &= \{[\, X \to Y\,]_2,\ [\, A \to \#\,]_2,\ [\, d \to \#\,]_2\} \\ &\cup \{[\, \alpha \to \#\,]_2 \mid \alpha \in U \cup \{a\}\}. \end{aligned}$$

3. We also consider the following tables:

$$\begin{aligned} R_{2,f} &= \{[\, f \to \lambda\,]_2,\ [\, d \to \#\,]_2\} \\ &\cup \{[\, \alpha \to \#\,]_2 \mid \alpha \in U \cup N_2\}, \\ R_{2,d} &= \{[\, d\,]_2 \to d,\ [\, a\,]_2 \to [\quad]_2 a',\ [\, \# \to \#\,]_2\}, \\ R_1 &= \{[\, a'\,]_1 \to [\quad]_1 a,\ [\, \# \to \#\,]_1\}. \end{aligned}$$

The equality $N(\Pi) = \{n \mid a^n \in L(G)\}$ follows in a similar way as in the previous proof, this time with the interplay of rules $[\ A\ ]_2 \to [\ \langle i \rangle\ ]_2 [\ d\ ]_2$ and $[\ e\ ]_2 \to [\ \#\ ]_2 [\ \#\ ]_2$ ensuring that the second rule of each matrix of type 2 or 4 is correctly simulated (used exactly once): if the second rule is used, then the computation never stops, hence $[\ A\ ]_2 \to [\ \langle i \rangle\ ]_2 [\ d\ ]_2$ must be used. In this way, membrane 2 is divided. In the first copy of the membrane we have the object $\langle i \rangle$, which will complete the simulation of the matrix. In the second copy of the membrane, the one containing the object d, we cannot use any table which contains the rule $d \to \#$, hence the only continuation is by using the table $R_{2,d}$. This dissolves the membrane, and its objects, remained free in the skin region, will no longer evolve. The matrices of type 3 are again simulated in only one step of a computation in Π. All copies of object a are immediately sent out of membrane 2 (to prevent their duplication when dividing the membrane), and from the skin region are sent out of the system. We leave the details to the reader and conclude that the system correctly simulates the matrix grammar G. □

One of the difficulties in the previous proofs was to inhibit the parallelism of using the rules of type (a_0). In membrane computing, the usual way to do this is by using *catalysts*, distinguished objects which never evolve, but can enter rules of the form $ca \to cv$, where a is a single object, which evolves under the control of the catalyst c. This idea can be considered also for P systems with active membranes, allowing rules of type (a_0) of the form $[\ ca \to cv\ ]_i$, where c is a catalyst, a is an object and v a multiset of objects. (When specifying a system with catalysts, the set C of catalysts is explicitly given after the set of objects.) We indicate the use of catalysts by writing cat_r in the notation for families of numbers computed by systems of a given type as above; r indicates the fact that at most r catalysts are used.

The previous results have the following counterpart for the catalytic case – with only two types of rules being used, and with only one membrane (note that one catalyst suffices).

Theorem 4.3 $NOP_1(tab, cat_1, a_0, c_0) = NRE$.

Proof. We consider again a matrix grammar with appearance checking $G = (N, \{a\}, S, M, F)$ in the binary normal form, with each terminal matrix $(X \to \lambda, A \to x)$ replaced with $(X \to f, A \to x)$, where f is a new symbol, and we construct the tabled P system with catalysts Π, with the components:

$$\begin{aligned} O &= N_1 \cup N_2 \cup \{a, c, d, f, \#\}, \\ C &= \{c\}, \\ \mu &= [\ \ ]_1, \\ w_1 &= X_0 A_0 d, \text{ where } (S \to X_0 A_0) \text{ is the initial matrix of } G, \end{aligned}$$

and the following tables.

1. For each matrix $m_i : (X \to Y, A \to x)$ in M of types 2 or 4, we consider the table

$$\begin{aligned} R_{1,i} &= \{[\ X \to Y\ ]_1,\ [\ cA \to cx\ ]_1,\ [\ cd \to c\#\ ]_1\} \\ &\cup \{[\ Z \to \#\ ]_1 \mid Z \in N_1 \cup \{f\}\}. \end{aligned}$$

2. For each matrix $m_i : (X \to Y, A \to \#)$ in M of type 3, we consider the table

$$\begin{aligned} R_{1,i} &= \{[\ X \to Y\ ]_1,\ [\ A \to \#\ ]_1\} \\ &\cup \{[\ Z \to \#\ ]_1 \mid Z \in N_1 \cup \{f\}\}. \end{aligned}$$

3. We also consider the following tables:

$$\begin{aligned} R_{1,f} &= \{[\ f \to \lambda\ ]_1,\ [\ d \to \lambda\ ]_1\} \\ &\cup \{[\ \alpha \to \#\ ]_1 \mid \alpha \in N_1 \cup N_2\}, \\ R_{1,a} &= \{[\ a\ ]_1 \to [\ \ ]_1 a,\ [\ \# \to \#\ ]_1\}. \end{aligned}$$

This time, the matrices m_i of types 2 and 4 are simulated by a single table, of type $R_{1,i}$: the first rule must be used (that is, the symbol X must be present), otherwise the trap-object is introduced; similarly, the rule $cA \to cx$ must be used, otherwise the catalyst will evolve together with the available object d and again the trap-object is introduced. Each matrix m_i of type 3 is simulated by the corresponding table $R_{1,i}$. After introducing the object f, no table as above can be used (without introducing the trap-object), hence we have to use $R_{1,f}$, which checks whether the derivation in G is terminal. At any time, the copies of object a are sent out by means of the table $R_{1,a}$. Consequently, $N(\Pi) = \{n \mid a^n \in L(G)\}$, and this completes the proof. □

The previous result is relevant in view of the fact that catalytic transition P systems with only one catalyst – not with active membranes – are not known to be universal, while the universality was proved for the case of using two catalysts [5].

5 Tables with Obligatory Rules

The idea of distinguishing some rules of each table and imposing that these rules are applied at least once when the tables are applied has at least two motivations. First, this is a way to also ensure the fact that a selected table does not leave unchanged the objects from the region where it is applied. Then, it reminds the matrices from matrix grammars, whose rules are all applied when applying a matrix. However, having several obligatory rules in the same table is a way to make the system cooperative: if both $a \to u$ and $b \to v$ must be simultaneously used at least once, then $ab \to uv$ must be used at least once (but the two cases are not equivalent, because besides evolving one a and one b, by rules $a \to u$ or $b \to v$ we can separately evolve further copies of a or of b, respectively).

That is why in what follows we allow at most one obligatory rule in each table. Such a rule is marked with a dot; when the table is used, its obligatory rule must be used at least once, otherwise the table is not allowed to be chosen.

This apparently small change in the definition of tabled P systems is powerful enough in order to lead to fast solutions (making use of membrane division) to computationally hard problems.

Theorem 5.1 *Tabled P systems with active membranes using obligatory rules (at most one in each table) can solve* `SAT` *in linear time; the construction is uniform, and the system is deterministic.*

Proof. Let us consider a propositional formula $\gamma = C_1 \wedge \ldots \wedge C_m$, consisting of m clauses $C_j = y_{j,1} \vee \ldots \vee y_{j,k_j}$, $1 \leq j \leq m$, where $y_{j,i} \in \{x_l, \neg x_l \mid 1 \leq l \leq n\}$, $1 \leq i \leq k_j$, (there are used n variables). Without loss of generality, we may assume that no clause contains two occurrences of some x_l or two occurrences of some $\neg x_l$ (the formula is not redundant at the level of clauses), or both x_l and $\neg x_l$ (otherwise such a clause is trivially satisfiable, hence can be removed).

We codify γ, which is an instance of SAT with size parameters n and m, by the multiset

$$\begin{aligned} w(\gamma) &= \{s_{j,i} \mid y_{j,r} = x_i, \text{ for some } 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq r \leq k_j\} \\ &\cup \{s'_{j,i} \mid y_{j,r} = \neg x_i, \text{ for some } 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq r \leq k_j\}. \end{aligned}$$

(We replace each variable x_i from each clause C_j with $s_{j,i}$ and each negated variable $\neg x_i$ from each clause C_j with $s'_{j,i}$, then we remove all parentheses and connectives. In this way we pass from γ to $w(\gamma)$ in a number of steps which is linear with respect to $n \cdot m$.)

We construct the P system Π with the following components:

$$\begin{aligned} O &= \{a_i \mid 1 \leq i \leq n+1\} \cup \{t_i, f_i \mid 1 \leq i \leq n\} \cup \{r_i, r'_i \mid 1 \leq i \leq m\} \\ &\cup \{d_i \mid 1 \leq i \leq m+1\} \cup \{c_i \mid 0 \leq i \leq 2n+m+3\} \\ &\cup \{s_{j,i}, s'_{j,i} \mid 1 \leq j \leq m, 1 \leq i \leq n\} \cup \{\texttt{yes}, \texttt{no}\}, \\ \mu &= [\,[\,[\;\;]_3\,]_2\,]_1, \\ w_1 &= \lambda, \\ w_2 &= c_0, \\ w_3 &= a_1, \\ R_{1,1} &= \{[\,\texttt{yes}\,]_1 \to [\;\;]_1\texttt{yes}, \\ &\quad [\,\texttt{no}\,]_1 \to [\;\;]_1\texttt{no}\}, \\ R_{2,1} &= \{[\,d_{m+1}\,]_2 \to \texttt{yes}, \\ &\quad [\,c_{2n+m+3}\,]_2 \to [\;\;]_2\texttt{no}\} \\ &\cup \{[\,c_i \to c_{i+1}\,]_2 \mid 0 \leq i \leq 2n+m+2\}, \\ R_{3,d} &= \{[\,a_i\,]_3 \to [\,t_i\,]_3[\,f_i\,]_3 \mid 1 \leq i \leq n\}, \\ R_{3,i,t} &= \{[\,t_i \xrightarrow{\bullet} a_{i+1}\} \\ &\cup \{[\,s_{j,i} \to r_j\,]_3 \mid 1 \leq j \leq m\}, \text{ for each } i = 1, 2, \ldots, n, \\ R_{3,i,f} &= \{[\,f_i \xrightarrow{\bullet} a_{i+1}\} \\ &\cup \{[\,s'_{j,i} \to r_j\,]_3 \mid 1 \leq j \leq m\}, \text{ for each } i = 1, 2, \ldots, n, \\ R_{3,0} &= \{[\,a_{n+1} \xrightarrow{\bullet} d_1\,]_3\} \\ &\cup \{[\,r_j \to r'_j\,]_3 \mid 1 \leq i \leq m\}, \\ R_{3,j} &= \{[\,r'_j \xrightarrow{\bullet} \lambda\,]_3\} \\ &\cup \{[\,d_i \to d_{i+1}\,]_3 \mid 1 \leq i \leq m\}, \text{ for each } j = 1, 2, \ldots, m, \\ R_{3,m+1} &= \{[\,d_{m+1}\,]_3 \to [\;\;]_3 d_{m+1}\}. \end{aligned}$$

There is no object in the skin membrane, while region 2 contains only the counter c_0, which will continuously increase its subscript, by means of table $R_{2,1}$. The "main work" is done in membrane 3. In the beginning, we have here the object a_1, hence the only applicable table is $R_{3,d}$, which divides the membrane, at the same time expanding the object a_1 to the truth values $t_1 = true$ and $f_1 = false$ of variable x_1. In the next step, the only tables which can be applied in the two membranes with label 3 are $R_{3,1,t}$ and $R_{3,1,f}$: the obligatory rules select the tables in a precise way. At the same time with the passage from t_1, f_1 to copies of a_2, we also introduce all clauses which are satisfied by t_1 and f_1, respectively. The process continues now with a_2, then with a_3, and so on, until expanding all variables and introducing all clauses satisfied by these truth assignments.

Therefore, after $2n$ steps we get 2^n membranes 3, containing the clauses satisfied by the 2^n possible truth assigments for the n variables.

In step $2n+1$ the only table which can be applied for membranes 3 is $R_{3,0}$: a_{n+1} is replaced with d_1 (which will check whether there is any membrane where all clauses are satisfied), and all r_j are primed.

From now on, for at most m steps, we use the tables $R_{3,j}$, $1 \leq j \leq m$. (Because these tables use primed versions of objects r_j, they were not applicable before using table $R_{3,0}$ – and this was the reason of priming.) Each of these tables removes the occurrences of one r'_j; because this operation is done by an obligatory rule, this is a way to check that the respective r'_j is present. At the same time, the subscript of the object d from each membrane 3 increases by one. If in a given membrane 3 there are copies of r'_j for all $j = 1, 2, \ldots, m$, then the respective object d reaches the subscript $m+1$, which indicates the fact that the corresponding truth-assignment has satisfied all clauses of γ. If a given membrane 3 does not contain copies of all r'_j, $1 \leq j \leq m$, then that membrane cannot evolve m steps, hence the local object d remains of the form d_j with $j \leq m$.

Simultaneously, the object from region 2 arrives at the form c_{2n+m+1}.

If at least one membrane 3 contains the object d_{m+1} (hence the formula is satisfiable), then in step $2n+m+2$ we use the table $R_{3,m+1}$ and the object d_{m+1} is sent to membrane 2 (at the same time, in region 2 we get c_{2n+m+2}). If no membrane 3 sends out the object d_{m+1}, hence the formula is not satisfiable, then the objects d_j with $j \leq m$ remain inside these membranes – but c_{2n+m+1} evolves to c_{2n+m+2} in region 2.

Now, in step $2n+m+3$, if any object d_{m+1} is present in region 2, then one of them will dissolve membrane 2, and will produce the object `yes`, which is left free in the skin region; in the next step, this object will leave the system, thus signaling that the formula is satisfiable. Because membrane 2 is dissolved, the object c_{2n+m+3} (obtained in step $2n+m+3$) also remains free in the skin region, where it cannot evolve any more. If no object d_{m+1} is present in region 2, then this membrane is not dissolved, c will get the subscript $2n+m+3$ and then in step $2n+m+4$ will exit membrane 2 transformed in `no`; in the next step, this object exits the systems, signaling that the formula is not satisfiable.

Thus, either we get `yes` outside the system in step $2n+m+4$, or `no` in step $2n+m+5$, and these objects correctly indicate whether or not γ is satisfiable.

The system Π can be constructed in polynomial time by a Turing machine, starting from n and m, and it works in a deterministic manner (after each reachable configuration there is at most one next configuration which can be correctly reached). □

If we are more interested in the time our system works than in the time of constructing it or in its deterministic behavior, then the answer to a given instance of `SAT` can be obtained in $n+m+4$ steps, by considering a system constructed in semi-uniform manner (starting directly from an instance of the problem) in the following way.

For a given formula γ as above, for $l = 1, 2, \ldots, n$, we denote

$$\begin{aligned} sat(t_l) &= \{r_j \mid \text{there is } 1 \leq i \leq k_j \text{ such that } y_{j,i} = x_l\}, \\ sat(f_l) &= \{r_j \mid \text{there is } 1 \leq i \leq k_j \text{ such that } y_{j,i} = \neg x_l\}. \end{aligned}$$

Then, we construct the system Π with (we omit specifying the set of objects, as well as several details about the way the system works, tasks which are left to the reader):

$$\begin{aligned} \mu &= [\,[\,[\quad]_3\,]_2\,]_1, \\ w_1 &= \lambda, \end{aligned}$$

$$
\begin{aligned}
w_2 &= c_0,\\
w_3 &= b_0 a_1 a_2 \ldots a_n,\\
R_{1,1} &= \{[\ \texttt{yes}\ ]_1 \to [\ \ ]_1\texttt{yes},\\
&\quad\ [\ \texttt{no}\ ]_1 \to [\ \ ]_1\texttt{no}\},\\
R_{2,1} &= \{[\ b_{n+m+2}\ ]_2 \to \texttt{yes},\\
&\quad\ [\ c_{n+m+3}\ ]_2 \to [\ \ ]_2\texttt{no}\}\\
&\cup\ \{[\ c_i \to c_{i+1}\ ]_2 \mid 0 \le i \le n+m+2\},\\
R_{3,i} &= \{[\ a_i\ ]_3 \xrightarrow{\bullet} [\ t_i\ ]_3 [\ f_i\ ]_3\}\\
&\cup\ \{[\ b_j \to b_{j+1}\ ]_3 \mid 0 \le j \le n-1\},\ \text{for each } i = 1, 2, \ldots, n,\\
R_{3,n+1} &= \{[\ b_n \xrightarrow{\bullet} b_{n+1}\ ]_3\}\\
&\cup\ \{[\ t_i \to sat(t_i)\ ]_3,\\
&\quad\ [\ f_i \to sat(f_i)\ ]_3 \mid 1 \le i \le n\},\\
R_{3,n+1+i} &= \{[\ r_i \xrightarrow{\bullet} \lambda\ ]_3\}\\
&\cup\ \{[\ b_{n+j} \to b_{n+j+1}\ ]_3 \mid 1 \le j \le m\},\ \text{for each } i = 1, 2, \ldots, m,\\
R_{3,n+m+2} &= \{[\ b_{n+m+1}\ ]_3 \to [\ \ ]_3 b_{n+m+2}\}.
\end{aligned}
$$

This time, in the first n steps we divide membrane 3 again and again, by means of the obligatory rules of tables $R_{3,i}$, $1 \le i \le n$, which expand the objects a_i to the truth values $t_i = true$ and $f_i = false$ of variable x_i. The order of using tables $R_{3,i}$ is arbitrary, but after n steps we get the same configuration irrespective of this order: 2^n membranes 3, containing the 2^n truth-assignments of the n variables, as well as the object b_n (at the same time, in membrane 2 we have obtained c_n).

In step $n+1$, in region 3 we can only apply $R_{3,n+1}$, which replaces b_n with b_{n+1} and each t_i, f_i by the clauses satisfied by these truth values (specifically, t_i is replaced by $sat(t_i)$ and f_i by $sat(f_i)$).

From now on, for at most m steps, we use the objects $b_{n+j+1}, 1 \le j \le m$, in the same way as objects d_j were used in the previous proof, in order to check whether or not at least one truth assignment has satisfied all clauses. If this is the case, then at least one membrane 3 will contain the object b_{n+m+1}, which will exit to membrane 2, will dissolve it in step $n+m+3$, and will produce the object `yes`, which then leave the system. If not, c_{n+m+3} will exit membrane 2 (in step $n+m+4$) transformed in `no`, which will exit the system in one further step.

The system Π can be constructed in polynomial time by a Turing machine, starting from γ (only the tables $R_{3,i}$ directly depend on the formula), and the system is clearly confluent.

6 Final Remarks

Contributing to the "campaign" of removing polarizations from P systems with active membranes, we have obtained several universality results for systems without polarizations, but having the rules structured in *tables*. When tables with (at most one) obligatory rules are used, **NP**-complete problems can be solved in linear time – this is illustrated with `SAT` problem.

Two important problems have remained open: (i) are systems without polarizations and without tables (maybe with catalysts) universal? (ii) can **NP**-complete problems be

solved in polynomial time by means of tabled P systems with active membranes without polarizations (and without using obligatory rules)?

Acknowledgements. The support of this research through the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds, is gratefully acknowledged.

References

[1] A. Alhazov, R. Freund, Gh. Păun, P systems with active membranes and two polarizations, in the present volume.

[2] A. Alhazov, L. Pan, Polarizationless P systems with active membranes, *Grammars*, 7, 1 (2004).

[3] A. Alhazov, L. Pan, Gh. Păun, Trading polarizations for labels in P systems with active membranes, submitted 2003.

[4] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.

[5] R. Freund, L. Kari, M. Oswald, P. Sosik, Computationally universal P systems without priorities: two catalysts are sufficient, submitted 2003.

[6] D. Hauschild, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, *Acta Informatica*, 31 (1994), 719–728.

[7] Gh. Păun, *Computing with Membranes: An Introduction*, Springer-Verlag, Berlin, 2002.

[8] M. Pérez-Jiménez, A. Roméro-Jimenez, F. Sancho-Caparrini, *Teoria de la complejidad en modelos de computación celular con membranas*, Kronos Editorial, Sevilla, 2002.

[9] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[10] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages* (3 volumes), Springer-Verlag, Berlin, 1997.

[11] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.