

A CHARACTERIZATION OF PARIKH SETS OF ETOL LANGUAGES IN TERMS OF P SYSTEMS

MASAMI ITO

*Department of Mathematics, Faculty of Science
Kyoto Sangyo University, Kyoto 603-8555, Japan
E-mail: ito@ksu.vx0.kyoto-su.ac.jp*

CARLOS MARTÍN-VIDE

*Research group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: cmv@astor.urv.es*

GHEORGHE PĂUN

*Institute of Mathematics of the Romanian Academy
PO Box 1-764, 70700 Bucureşti, Romania
E-mail: gpaun@imar.ro*

We prove that the Parikh sets of ETOL languages are exactly the sets of vectors of natural numbers computed by P systems (without cooperating rules, without priorities, and without target indications) which can create new membranes as a result of objects evolution (at any moment of a computation, the number of membranes is bounded by a given constant; two membranes at any moment are sufficient).

1 Introduction

P systems are a class of distributed parallel computing models introduced in [4], inspired from the way the alive cells process chemical compounds, energy, and information. In short, in the *regions* delimited by a *membrane structure* (see Figure 1 for an illustration of this notion), one places *multisets* of *objects*, which evolve according to *evolution rules* associated with the regions; a *computation* consists of transitions among system *configurations*; the *result* of a halting computation is the vector of the multiplicities of objects present in the final configuration in a specified *output membrane* or of objects which leave the external membrane of the system (the *skin membrane*) during a computation. That is, a P system *computes* a set of vectors of natural numbers. Many variants characterize the family of recursively enumerable sets of vectors of natural numbers, which are exactly the Parikh sets associated with recursively enumerable languages. Details can be found in [4], [1], etc. When membrane division is allowed, NP-complete

problems can be solved in linear time, [7], [3]. (The current bibliography of the domain, as well as many downloadable papers, can be found at the web address <http://bioinformatics.bio.disco.unimib.it/psystems>.)

In [2] it is proved that the vectors computed by P systems without cooperating rules and without priorities are Parikh sets of ET0L languages, but the converse is left open. We do not solve here this question (we do not necessarily believe that the converse is even true), but we consider a slight modification of P systems which is able to lead to a characterization of Parikh sets of ET0L languages. Namely, we add the possibility that a membrane can be created as a result of an object evolution (this supposes that a possible list of membranes are given, so that the rules which can act in the region of the new membrane are identified by the label of the membrane; always a new membrane is an elementary one). The P systems with this feature, without cooperating rules and without using a priority relation among its rules and with the communication of objects controlled by indications *here*, *out*, *in* (hence without using the powerful indication in_j , which also specifies the label, j , of the target membrane), and which at any time have a number of membranes bounded by a given constant, exactly generate the Parikh sets of ET0L languages; specifically, two membranes suffice (but we need membranes of three types).

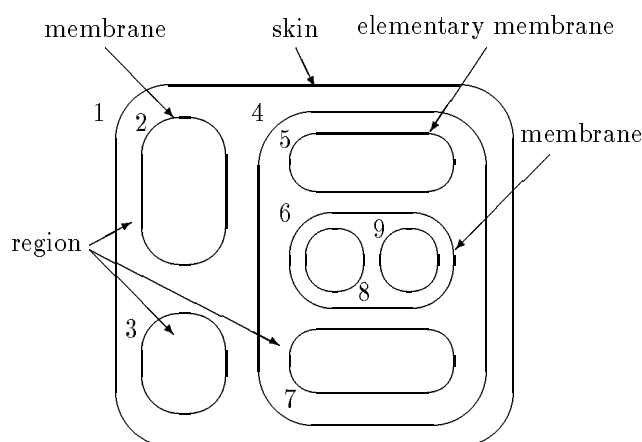


Figure 1: A membrane structure

We do not know whether or not systems without a bound on the number of membranes can generate sets of vectors which are not Parikh sets of ET0L languages. (In order to positively solve this question we would first need to

know sets of vectors of natural numbers which are not the Parikh sets of ETOL languages, and such examples are not at all frequent in the “classic” L systems theory; for instance, from [9] we find only one example, $\{a^n \mid n \text{ a prime number}\}$.)

2 P Systems with Membrane Creation

We refer to [10] for the elements of formal language theory we use here. We only specify that for a string $x \in V^*$ and a symbol $a \in V$, we denote by $|x|$ the length of x and by $|x|_a$ the number of occurrences of the symbol a in the string x . The families of context-free, context-sensitive, and recursively enumerable languages are denoted by CF, CS, RE , respectively. For $w \in V^*$, $V = \{a_1, \dots, a_n\}$, we denote by $\Psi_V(w)$ the Parikh vector of w , that is, $\Psi_V(w) = (|w|_{a_1}, \dots, |w|_{a_n})$; this is extended to languages in the natural way. For a family FL of languages, we denote by $PsFL$ the family of Parikh sets of vectors associated with languages in FL .

A membrane structure will be represented by a string of matching labeled parentheses. For instance, the membrane structure in Figure 1 is represented by

$$[{}_1 [{}_2]_2 [{}_3]_3 [{}_4 [{}_5]_5 [{}_6 [{}_8]_8 [{}_9]_9]_6 [{}_7]_7]_4]_1.$$

A multiset over an alphabet V is represented by a string over V and each string precisely identifies a multiset; the Parikh vector associated with the string indicates the multiplicities of each element of V in the corresponding multiset. Thus, when speaking of a “multiset” $w \in V^*$ we understand the multiset identified by w .

We are now ready to introduce the class of P systems we will investigate in this paper, and we introduce it in the particular variant we will consider below.

A *P system with membrane creation* is a construct

$$\Pi = (V, T, \mu, w_1, \dots, w_k, R_1, \dots, R_n),$$

where:

- (i) V is an alphabet; its elements are called *objects*;
- (ii) $T \subseteq V$ (the *output* alphabet);
- (iii) μ is a membrane structure consisting of k membranes, with the membranes and the regions labeled (not necessarily in a one-to-one manner) by elements of a given set Λ ; let us assume that the possible labels are $1, 2, \dots, n$, and that the skin membrane and only this membrane is labeled with 1; then, the membranes in μ are labeled by i_1, i_2, \dots, i_k , for some $1 \leq i_j \leq n, 1 \leq j \leq k$, with $i_1 = 1$;

- (iv) w_1, \dots, w_k , are multisets over V associated with the regions i_1, i_2, \dots, i_k of μ ;
- (v) $R_i, 1 \leq i \leq n$, are finite sets of *evolution rules* over V . These rules are of the form $a \rightarrow v, a \rightarrow v\delta$, or $a \rightarrow [{}_i v]_i$, where $a \in V, 1 \leq i \leq n, v$ is a string over

$$V \times \{here, out, in\}$$

and δ is a special symbol not in V . In R_1 no rule can contain δ and no rule of the form $a \rightarrow [{}_1 v]_1$ can appear in any set R_i .

When presenting the evolution rules, the indication “here” is in general omitted.

The membrane structure and the multisets in Π constitute the *initial configuration* of the system. Note that the initial configuration can contain a number k of membranes which has no relation with n , the number of possible *types* of membranes. The rules in a set R_i are applicable to objects in each region delimited by a membrane with the label i (that is, the identification between regions and sets of rules is given by the labels of membranes, which appear as subscripts of sets of rules). The passing from a configuration of the system to another configuration is done by a maximal parallel application of rules: all objects, from all membranes, which can be subject of local evolution rules should evolve by means of these rules. The choice of rules to be used and of objects to which these rules are applied is done in a nondeterministic way.

The application of a rule $a \rightarrow v$ in a region containing a multiset w means to remove a copy of the object a from w and to add the objects specified by v , following the prescriptions given by v : If an object appears in v in the form $(a, here)$, then it remains in the same region; if it appears in the form (a, out) , then a copy of the object a will be introduced in the region of the membrane placed directly outside the region of the rule $u \rightarrow v$ (if the rule is applied in the skin membrane, then a is sent out of the system); if it appears in the form (a, in) , then a copy of a is introduced in one of the membranes placed directly inside the region of the rule $a \rightarrow v$, nondeterministically chosen, if such a membrane exists, otherwise the rule cannot be applied. If the special symbol δ appears, then the membrane which delimits the region where we work is dissolved, and all the objects in this region become elements of the region placed immediately outside, while the rules of the dissolved membrane are removed. When applying a rule $a \rightarrow [{}_i v]_i$ in a region j , a copy of a is removed and a membrane with the label i is created, containing the multiset v , inside the region of membrane j . In this new membrane, the set R_i of rules will be applied. In this way, several membranes with the label i (and hence the same rules) can be present in various

places of the membrane structure of the system. The skin membrane is never dissolved and we can never create a new membrane with label 1.

A sequence of transitions between configurations of Π is called a *computation* of Π . A computation is *successful* if and only if it halts, that is, there is no rule applicable to the objects present in the last configuration. The result of a successful computation is $\Psi_T(w)$, where w describes the multiset of objects from T which have left the skin membrane during the computation; we say that this vector is *generated* by Π . (Note that we take into account only the objects from T .) We denote by $N(\Pi)$ the set of all vectors generated by Π and by $N_m(\Pi)$ the set of vectors generated by computations whose configurations contain at most m membranes, for a given $m \geq 1$.

The family of all sets $N(\Pi)$ is denoted by $CP(\delta)$; if only systems without using the membrane dissolving action are used, then we replace δ by $n\delta$. For each $m \geq 1$, the family of all sets $N_m(\Pi)$ is denoted by $CP_m(\alpha)$ and their union over m is denoted by $CP_*(\alpha)$, $\alpha \in \{\delta, n\delta\}$. When the membrane creation is not used, we remove the letter C from the front of notations of these families.

We close this section with an *example*: let us consider the P system

$$\Pi = (\{a, a', b, c\}, \{b\}, [{}_1[{}_2]_2]_1, \lambda, a'c, R_1, R_2, R_3),$$

with the following sets of rules:

$$\begin{aligned} R_1 &= \{a \rightarrow a', a' \rightarrow (a, in), c \rightarrow [{}_3c]_3, b \rightarrow (a, out)\}, \\ R_2 &= \{a' \rightarrow a, a \rightarrow a^2, c \rightarrow c, c \rightarrow c\delta\}, \\ R_3 &= \{a \rightarrow b, b \rightarrow b^3, c \rightarrow c, c \rightarrow \delta\}. \end{aligned}$$

Initially, we have only two membranes, with only two objects, in membrane 2, a copy of a' and one of c . We produce 2^n copies of a , for some $n \geq 0$, in $n + 1$ steps; in the first step we pass from a' to a , in the last one the membrane is dissolved and 2^n copies of a and one copy of c are left free in membrane 1. In the next step, each a is replaced by a' and, simultaneously, a copy of membrane 3 is produced. All objects are sent to this membrane, where first we replace a by b , then we multiply by 3, repeatedly, the number of objects b . At any moment, also membrane 3 is dissolved. At the next step, all copies of b are sent out of the system.

Consequently, $N(\Pi) = \{(2^n 3^m) \mid n, m \geq 0\}$. Note that we always have at most two membranes in our system, hence $\bar{N}(\Pi) = N_2(\Pi)$.

3 Some Preliminary Remarks

We mention here some results about the generative capacity of the P systems with membrane creating possibilities which either directly follow from the definitions or are consequences of results proved in [4], [2], [8]. Note that our systems are more general than those in the mentioned papers, in the sense that if we skip the membrane creation feature we get a system as in those papers. A slight difference appears in what concerns the way of defining the result of a computation. Here we consider the multiset of objects leaving the system, while in [4], [2] one deals with the multiset of objects present in a specified output membrane at the end of a computation, and in [8] one considers the strings of objects which leave the system, arranged in the order they exit from the skin membrane. However, these differences are not important in what concerns the generative power, systems with the output defined in one way can be simulated by systems with the output defined in another way (sometimes, one further membrane is necessary). In our case, we have chosen to read the result of a computation outside the system because the membrane structure can dramatically change its shape by dissolving and creating membranes; moreover, we do not have a one-to-one labeling of membranes, thus we cannot indicate in advance an output membrane.

In what follows, we need the notion of an *ETOL system*, which is a construct $G = (V, T, w, P_1, \dots, P_m)$, $m \geq 1$, where V is an alphabet, $T \subseteq V$, $w \in V^*$, and $P_i, 1 \leq i \leq m$, are finite sets (*tables*) of context-free rules over V such that for each $a \in V$ there is at least one rule $a \rightarrow x$ in each set P_i (we say that these tables are *complete*). In a derivation step, all the symbols present in the current sentential form are rewritten using *one* table. The language generated by G , denoted by $L(G)$, consists of all strings over T which can be generated in this way, starting from w . An ETOL system with only one table is called an *EOL system*. Details can be found, e.g., in [9].

We denote by *EOL* and *ETOL* the families of languages generated by EOL and ETOL systems, respectively.

The following inclusions are known:

$$PsCF \subset PsEOL \subset PsETOL \subset PsCS.$$

For instance, $\{(2^n) \mid n \geq 1\} \in PsEOL - PsCF$, $\{(2^n 3^m) \mid n, m \geq 0\} \in PsETOL - PsEOL$ (according to Exercise II.4.4 from [9], $\{a^{2^n 3^m} \mid n, m \geq 0\} \in ETOL - EOL$; note that this set is computed by the P system in the example from the previous section, hence we get the fact that $CP_2(nPri, \delta) - PsEOL \neq \emptyset$), and $\{(n) \mid n \text{ prime}\} \in PsCS - PsETOL$ (see Exercise VI.2.6 in [9]).

Using these relations, from the definitions and from [4], [2], [8], we obtain the following relations:

- (i) $PsCF = CP_1(n\delta) \subset PsE0L \subseteq CP_2(\delta)$.
- (ii) $P_i(\alpha) \subseteq CP_i(\alpha)$, $\alpha \in \{\delta, n\delta\}$ for all $i \geq 1$.
- (iii) $P_i(\alpha) \subseteq P_{i+1}(\alpha)$ for all $i \geq 1$ $\alpha \in \{\delta, n\delta\}$.

If catalysts are used, hence (non-context-free) rules of the form $ca \rightarrow cv$, where c is an object which only assists other objects when evolving and it is not changed during a computation, and also a priority relation among rules is considered, then a characterization of Parikh sets of recursively enumerable languages is obtained.

In [2] it is proved that the Parikh sets of ET0L languages can be generated by P systems with priorities and without using the membrane dissolving action (Theorem 6) and that $P_*(\delta) \subseteq PsET0L$ (Theorem 7), but the relations between $PsET0L$ and $P_i(\alpha)$, $i \geq 1$, $\alpha \in \{\delta, n\delta\}$, are not settled. In the case of using the membrane creation feature, the inclusion $PsET0L \subseteq CP_i(\delta)$ is easy to be obtained, even for a small value of i , namely, $i = 2$, as we shall immediately see.

4 Characterizing $PsET0L$

We summarize the main result of this paper in the next theorem, then we give its proof in two lemmas:

Theorem 1. $PsET0L = CP_m(\delta) = CP_*(\delta)$, for all $m \geq 2$.

Lemma 1. $PsET0L \subseteq CP_2(\delta)$.

Proof. According to Theorem 1.3 in [9], for each language $L \in ET0L$ there is an ET0L system G which generates L and contains only two tables, that is, $G = (V, T, w, P_1, P_2)$. Moreover, if we examine the proof of that theorem, we find that after each use of table P_1 we either use again table P_1 or we use table P_2 , but after each use of table P_2 we always use table P_1 ; at the first step of a derivation, we use table P_1 . Making use of this observation, we construct a P system as follows.

Denote $V' = \{a' \mid a \in V\}$, $V'' = \{a'' \mid a \in V\}$ and define the morphism h by $h(a) = a'$, $a \in V$. Then, let us consider the system

$$\Pi = (V \cup V' \cup V'' \cup T \cup \{c\}, T, [{}_1[{}_2]_2]_1, \lambda, wc, R_1, R_2, R_3),$$

with the following sets of rules:

$$R_1 = \{a \rightarrow h(v) \mid a \rightarrow v \in P_2\} \\ \cup \{a' \rightarrow (a, in) \mid a \in V\}$$

$$\begin{aligned}
& \cup \{a'' \rightarrow (a, out) \mid a \in T\} \\
& \cup \{a'' \rightarrow a'' \mid a \in V - T\} \\
& \cup \{c \rightarrow [{}_2c]_2, c \rightarrow [{}_3c]_3\}, \\
R_2 &= P_1 \cup \{c \rightarrow c, c \rightarrow c\delta\}, \\
R_3 &= \{a \rightarrow a'' \mid a \in V\} \\
& \cup \{c \rightarrow \delta\}.
\end{aligned}$$

The system works as follows. In the initial configuration we have only two membranes, $\mu = [{}_1[{}_2]_2]_1$. Table P_1 is simulated in membrane 2 any number of times; when the rule $c \rightarrow c\delta$ is used, the membrane is dissolved and its contents is left free in membrane 1. We simulate here the use of table P_2 (all symbols are primed during this simulation) and, at the same time, either a membrane 2 or a membrane 3 is created by c . At the next step, all objects are sent to the newly created membranes (without primes). If the created membrane was 2, then the process can be iterated. In this way, we can simulate any derivation in G , and this is correct, because of the parallel mode of using the rules in the ET0L system G and the P system Π and because of the way of using the tables of G (always after using P_2 we pass to using at least once P_1).

If the membrane with label 3 is introduced, then all objects sent to it are doubly primed and at the same time the membrane is dissolved. In the skin membrane, the double primed symbols are either sent out – providing that they are from T – or can evolve for ever – in the case that they are from $V - T$. In this way, we check whether or not the derivation in G was terminal; in the latter case, the computation in Π never stops.

Consequently, $N(\Pi) = Ps(L(G))$. Because we always have at most two membranes present in our system, the proof is complete. \square

Note that although at any moment we have at most two membranes present in the current configurations of our system, these membranes can be of three types. We do not believe that the number of types can also be decreased to two.

We pass now to the difficult part of the proof, an extension of the technique from the proof of Theorem 7 from [2] to the case when also the membrane creating feature is present.

Lemma 2. $CP_*(\delta) \subseteq PsET0L$.

Proof. Let us consider a P system $\Pi = (V, T, \mu, w_1, \dots, w_k, R_1, \dots, R_n)$, for some $k \geq 1$ and $n \geq 1$. Our goal is to construct an ET0L system G such that $\Psi_T(L(G)) = N_m(\Pi)$, for some given $m \geq 1$. To this aim, the following notations are useful.

In order to distinguish the copies of the same membrane which may be simultaneously present in a configuration, we will label them with pairs (i, j) of integers, such that $1 \leq i \leq n$ identifies the type of the membrane and $1 \leq j \leq m$ identifies the copy of the membrane of type i . Always, the skin membrane will be labeled with $(1, 1)$ and only one copy of this membrane is present.

Consider all symbols $e_{i,j}^{i',j'}$, for $1 \leq i, i' \leq n$ and $1 \leq j, j' \leq m$, plus the symbol $e_{1,1}^{\infty,\infty}$. These symbols identify each membrane by the pair (i, j) and the membrane directly containing membrane (i, j) ; in the case of the skin membrane, we consider the outside region, identified by (∞, ∞) , as the covering "membrane".

We denote by E the set of all symbols of this form.

Consider now the language F over E consisting of all strings u such that $|u| \leq m$, associated with all possible membrane structures consisting of at most m membranes, labeled with (i, j) as above, such that the labels are distinct (if two copies of a membrane of the same type i are present, then one is labeled with (i, j) and the other with (i, j') , for $j \neq j'$). For instance, for the membrane structure described by the string

$$[_{(1,1)}[_{(2,1)}]_{(2,1)}]_{(3,2)}[_{(3,2)}[_{(4,3)}[_{(2,2)}]_{(2,2)}]_{(4,3)}]_{(1,1)}$$

a possible string of this type is

$$e_{2,1}^{1,1} e_{1,1}^{\infty,\infty} e_{4,3}^{1,1} e_{3,2}^{1,1} e_{2,2}^{4,3}.$$

Note that any permutation of the same string in F determines the same membrane structure and, indeed, such a string precisely identifies a membrane structure, because the subscripts and the superscripts of symbols in E correspond to edges in the tree associated with the membrane structure. For each $u \in F$ we denote by \bar{u} the class of all permutations of u ; let \bar{F} be the set of all classes \bar{u} , for $u \in F$.

Note also that not all strings in E^* of length at most m are in F ; for instance, if $e_{i,j}^{i',j'}$ appears in a string, and $(i', j') \neq (\infty, \infty)$, then also $e_{i',j'}^{i'',j''}$ should appear, for some i'', j'' .

Consider also the sets

$$\begin{aligned} C &= \{c_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}, \\ C' &= \{c'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}, \\ D &= \{d_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}, \\ D' &= \{d'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}. \end{aligned}$$

The elements of C and D identify all possible membranes in a membrane structure. When some $c_{i,j}$ (resp., $d_{i,j}$) will be replaced by $c'_{i,j}$ (resp., $d'_{i,j}$), this

will indicate the fact that a membrane with the label (i, j) was created (resp., dissolved).

Let $u \in F$ describe a membrane structure μ_u and let $v \in D'^*$ be a string such that for each $d'_{i,j}$ appearing in v , the symbol $e_{i,j}^{i',j'}$ appears in u (we say that v identifies a substructure of u). Let us interpret the string v as identifying membranes of μ_u which are dissolved. The membrane structure obtained from the membrane structure described by u by dissolving the membranes identified by v is described by a string z in F ; we denote the class \bar{z} by $f(u, v)$ (that is, f is an operator which associates with any membrane structure and with any set of dissolved membranes from this structure a description of the resulting membrane structure). Clearly, f is a computable operator.

Consider now the objects appearing in the regions of Π . If an object $a \in V$ is present in the region of a membrane labeled with (i, j) , then we also mark the object with (i, j) , and write $a^{(i,j)}$.

For $u \in F$, $v \in D'^*$ such that v identifies a substructure of u , and for (i, j) such that there is a symbol $e_{i,j}^{i',j'}$ in u , we denote by $g(i, j; u, v)$ the pair (i'', j'') of integers which identify the region of the membrane structure identified by $f(u, v)$ where the objects from membrane (i, j) of u will be placed after dissolving the membranes identified by v . Clearly, $g(i, j; u, \lambda) = (i, j)$.

Note that knowing u and v we precisely know the place of all objects from the membrane structure described by u after dissolving the membranes indicated by v , hence also g is a computable operator.

Let u_0 be the string from F which identifies the initial membrane structure, μ , of Π , let w_0 be the concatenation of the strings describing the initial multisets of Π , after replacing each object a by $a^{(i,j)}$, according to the region (i, j) where a is placed, let z_C, z_D be the concatenation of all symbols from sets C, D , respectively.

We now pass to define the ET0L system G we look for.

Its total alphabet is

$$T \cup C \cup C' \cup D \cup D' \cup \{\bar{u} \mid u \in F\} \cup \{X, \#\} \\ \cup \{a^{(i,j)} \mid a \in V, 1 \leq i \leq n, 1 \leq j \leq m\},$$

the terminal alphabet is T (X and $\#$ are new symbols; $\#$ is a trap-symbol, which can never be removed), the axiom is

$$Xw_0[\bar{u}_0]z_Cz_D,$$

while the set of tables is constructed as follows (for each table we specify only the rules of interest for our proof, but not the completion rules for symbols α for which no rule of the form $\alpha \rightarrow x$ is already given; that is, rules of the form $\alpha \rightarrow \alpha$ should be added to all tables when necessary).

1. For each $u \in F$ we consider the following table:

$$\begin{aligned}
P_u = & \{X \rightarrow X', X' \rightarrow \#, [\bar{u}] \rightarrow [\bar{u}]\} \\
& \cup \{[\bar{u}'] \rightarrow \# \mid \bar{u}' \in \bar{F} - \{\bar{u}\}\} \\
& \cup \{\alpha \rightarrow \# \mid \text{for all } \alpha \text{ in } D' \cup C'\} \\
& \cup \{a^{(i,j)} \rightarrow v' \mid 1 \leq i \leq n, 1 \leq j \leq m, \\
& \quad a \rightarrow v \in R_i, e_{i,j}^{i',j'} \text{ appears in } u \text{ for some } i', j', \\
& \quad \text{and } v' \text{ is obtained from } v \text{ in the following way:} \\
& \quad \text{if } (b, \textit{here}) \text{ appears in } v, \text{ then we put } b^{(i,j)} \text{ in } v', \\
& \quad \text{if } (b, \textit{out}) \text{ appears in } v, \text{ then we put } b^{(i',j')} \text{ in } v', \\
& \quad \text{if } (b, \textit{in}) \text{ appears in } v, \text{ then we put } b^{(i'',j'')} \text{ in } v' \\
& \quad \quad \text{for some } (i'', j'') \text{ associated with a membrane} \\
& \quad \quad \text{which is directly inside membrane } (i, j), \\
& \quad \text{but if } (b, \textit{out}) \text{ appears in } v \text{ and we have } (i, j) = (1, 1), \\
& \quad \text{then we put } b \text{ in } v' \text{ if } b \in T, \\
& \quad \text{and we replace it by } \lambda \text{ if } b \in V - T\} \\
& \cup \{a^{(i,j)} \rightarrow v', d_{i,j} \rightarrow d'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m, \\
& \quad a \rightarrow v\delta \in R_i, e_{i,j}^{i',j'} \text{ appears in } u \text{ for some } i', j', \\
& \quad \text{and } v' \text{ is obtained from } v \text{ in the same way as above}\} \\
& \cup \{a^{(i,j)} \rightarrow v^{(k,l)} e_{k,l}^{i,j}, c_{k,l} \rightarrow c'_{k,l} \mid 1 \leq i \leq n, 1 \leq j \leq m, \\
& \quad a \rightarrow [{}_k v]_k \in R_i, \text{ for some } (k, l) \text{ which does not appear} \\
& \quad \text{as a superscript in } u \text{ and } v^{(k,l)} \text{ is obtained by replacing} \\
& \quad \text{each symbol } b \text{ which appears in } v \text{ by the symbol } b^{(k,l)}\}.
\end{aligned}$$

2. For each $u \in F, z \in D'^*$, and $y \in C'^*$ such that $f(u, z)y \in F$ we consider the following table (u indicates existing membranes, z indicates membranes of u which were dissolved at the previous step, and y indicates membranes which were created, hence they are added to membranes of u after removing the membranes indicated by z):

$$\begin{aligned}
P_{u,z,y} = & \{X' \rightarrow X, X \rightarrow \#, [\bar{u}] \rightarrow [\overline{f(u, z)y}]\} \\
& \cup \{[\bar{u}'] \rightarrow \# \mid \bar{u}' \in \bar{F} - \{\bar{u}\}\} \\
& \cup \{a^{(i,j)} \rightarrow a^{g(i,j;u,z)} \mid 1 \leq i \leq n, 1 \leq j \leq m, a \in V, \\
& \quad \text{for each } (i, j) \text{ which appears as a subscript in } u\}
\end{aligned}$$

$$\begin{aligned}
& \cup \{d'_{i,j} \rightarrow d_{i,j}, d_{i,j} \rightarrow \# \mid \text{for } d'_{i,j} \text{ appearing in } z\} \\
& \cup \{d'_{i,j} \rightarrow \# \mid \text{for } d'_{i,j} \text{ not appearing in } z\} \\
& \cup \{c'_{i,j} \rightarrow c_{i,j}, c_{i,j} \rightarrow \# \mid \text{for } c'_{i,j} \text{ appearing in } y\} \\
& \cup \{c'_{i,j} \rightarrow \# \mid \text{for } c'_{i,j} \text{ not appearing in } y\}.
\end{aligned}$$

3. For each $u \in F$ we consider the following table:

$$\begin{aligned}
P'_u = & \{X \rightarrow \lambda, X' \rightarrow \#, [\bar{u}] \rightarrow \lambda\} \\
& \cup \{[\bar{u}'] \rightarrow \# \mid \bar{u}' \in \bar{F} - \{\bar{u}\}\} \\
& \cup \{\alpha \rightarrow \# \mid \text{for all } \alpha \text{ in } C' \cup D'\} \\
& \cup \{a^{(i,j)} \rightarrow a \mid (i, j) \text{ appears as a subscript in } u \\
& \quad \text{and no rule } a \rightarrow v \text{ exists in } R_i\} \\
& \cup \{a^{(i,j)} \rightarrow \# \mid (i, j) \text{ appears as a subscript in } u \\
& \quad \text{and there is a rule } a \rightarrow v \text{ in } R_i\} \\
& \cup \{d_{i,j} \rightarrow \lambda, c_{i,j} \rightarrow \lambda \mid 1 \leq i \leq n, 1 \leq j \leq m\}.
\end{aligned}$$

This ET0L system works as follows.

The nonterminal symbols $[\bar{u}]$, for $u \in F$, precisely identify the membrane structure. The symbols in D and C are associated with all possible copies of membranes, while the symbols in D' and C' indicate the membranes which are deleted, respectively created, at a given step of a computation. Initially, all symbols from D and C are present in the axiom of G , together with a description of the initial membrane structure and of the initial multiset. It is also present the control symbol X .

In the presence of X only tables of the form P_u, P'_u can be used (if a rule $X' \rightarrow \#$ is used, then the derivation will never be a terminal one, because $\#$ can never be eliminated).

The tables of the form P_u simulate the transitions in the P system Π . Because each object a present in the region of a membrane (i, j) is present in G in the form $a^{(i,j)}$, we precisely know the rules which can be applied to this object, namely, those from R_i . The application of such rules is simulated in such a way to take care of the communication commands *here*, *out*, *in*, the dissolving actions, and the membrane creation. Note that the string u contains the necessary information for correctly handling the indication *in*: we know which are the membranes placed inside the membrane where we work, hence we can choose one of them, nondeterministically. The membranes which are dissolved are identified with symbols from D' and the membranes which are created are identified with symbols from C' .

At the next step, we have to use a table of the type $P_{u,z,y}$, for $u \in F$ identifying the existing membranes, $z \in D'^*$ identifying the dissolved membranes, and $y \in C'^*$ identifying the created membranes. This table will both check whether or not the strings z, y indicate exactly the dissolved and created membranes, indeed (in the negative case, the trap-symbol $\#$ will be introduced) and it also computes the new string in F identifying the current membrane structure, as well as the new superscripts (i', j') of objects. This is done by means of the operators f and g defined at the beginning of the proof.

Because X' is again replaced by X , we can iterate this procedure, hence any computation in Π can be simulated by a derivation in G .

At any moment, we can use a “terminal” table P'_u , for some $u \in F$. It erases both the description u of the membrane structure and the symbol X , hence no further derivation step is possible. We check now also whether or not we have reached a halting configuration in Π : if any further rule can be applied in Π to the multisets corresponding to the obtained sentential form of G , then the trap-symbol $\#$ is introduced. All nonterminals in $D \cup C$ are removed.

If the derivation is terminal, that is, no occurrence of $\#$ is present in the string x obtained in this way, then $\Psi_T(x)$ is exactly the vector of multiplicities describing the multiset computed by Π : each object $a \in T$ which is sent out of the skin membrane during the computation in Π is introduced by a table of type P_u in the form a , and the only rules which process such a symbol are the completion rules $a \rightarrow a$; when a symbol $a \in V - T$ is to be sent out of the skin membrane of Π , this symbol is simply erased, hence it does not appear in the final string generated by G .

In conclusion, $\Psi_T(L(G)) = N(\Pi)$. □

The reader can check that we can work also with types of P systems different from that considered above and the inclusions in Lemmas 1 and 2 still hold. For instance, we can consider target indications of the form in_j : when (a, in_j) appears in the right hand of a rule $b \rightarrow v$ which is applied in a membrane i , then a copy of a is sent to the membrane with the label j , providing that it appears immediately inside membrane i . Such a communication feature was considered in [4] and in many other papers. Note that this is a much stronger communication command than in , where the object is sent to one of the lower membranes, nondeterministically choosing it. Another possibility is to use *electrical charges* associated with objects and membranes, as proposed in [6]. Because in Lemma 1 we have only one internal membrane and because the construction in Lemma 2 can handle the electrical charges (which are markings $+, -, 0$), again the result in Theorem 1 holds also for this variant.

5 Final Remarks

Up to now, most results about various classes of P systems have provided characterizations of recursively enumerable sets of vectors of natural numbers; for some very particular cases, characterizations of *PsCF* were easily obtained. The result proved above is the first one giving a characterization of the Parikh sets of languages in a family different from *CF* and *RE*. The fact that this family is *ETOL* is not unexpected: both P systems and ETOL systems use context-free rules, applied in a parallel manner, both of them use auxiliary symbols (this is not an essential feature in P systems, but it is crucial for ETOL systems). It would be of interest to also find characterizations by P systems of Parikh sets of other families of languages in the Chomsky or Lindenmayer areas. In particular, it is of interest to get a characterization of Parikh sets of deterministic ETOL languages (to this aim it is probably necessary to consider a sort of determinism also for P systems).

Another important problem is to make use of the enhanced parallelism provided by the membrane creation feature in order to solve hard problems in a feasible time (trading space for time). For instance, by a rule $a \rightarrow aa$, in n steps we can generate 2^n copies of the object a , then, by a rule $a \rightarrow [{}_i v]_i$ we can create 2^n copies of the same membrane. By using exponentially many membranes, in [7] and [3] one solves NP-complete problems in linear time. Maybe this is possible also for P systems with membrane creation features; the difficulty seems to be the fact that all copies of the membrane i introduced by a rule $a \rightarrow [{}_i v]_i$ as above contain the same objects, those identified by v (in [7], [3] we have much more freedom from this point of view).

Note. The work of Gh. Păun was partially supported by a grant of NATO Science Committee, Spain, 2000–2001.

References

1. C. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor and Francis, London, 2000 (Chapter 3: “Computing with Membranes”).
2. J. Dassow, Gh. Păun, On the power of membrane computing, *J. of Universal Computer Sci.*, **5**, 2 (1999), 33–49 (www.iicm.edu/jucs).
3. S. N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP-complete problems, *Romanian J. of Information Science and Technology*, **2**, 4 (1999), 357–367.
4. Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, **61**, 1 (2000), 108–143 (paper circulated in a preliminary form)

- as *Turku Center for Computer Science Research Report* No 208, November 1998).
5. Gh. Păun, Computing with membranes. An introduction, *Bulletin of the EATCS*, **67** (1999), 139–152.
 6. Gh. Păun, Computing with membranes – A variant: P systems with polarized membranes, *Intern. J. of Foundations of Computer Science*, **11**, 1 (2000), 167–182.
 7. Gh. Păun, P systems with active membranes: Attacking NP-complete problems, *J. Automat, Languages and Combinatorics*, **6**, 1 (2001), 75–90.
 8. Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae*, **41**, 3 (2000), 259–266.
 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*, Springer-Verlag, Heidelberg, 1997.