On the Power of P Systems with Replicated Rewriting

Vincenzo MANCA
Università degli Studi di Pisa, Dipartimento di Informatica
Corso Italia 40, 56125 Pisa, Italy
E-mail: mancav@di.unipi.it

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

Abstract

We continue the study of a variant of string-objects P systems recently introduced by Krishna and Rama, namely with the possibility to also replicate the strings when rewriting them. The main result of the paper solves an open problem about such P systems: the hierarchy on the number of membranes collapses, systems with six membranes characterize the recursively enumerable languages. We also investigate the power of systems with less than six membranes, in comparison with the families of matrix languages and of E0L and ET0L languages. We close the paper with some remarks about the closure properties of the families of languages generated by P systems with replicated rewriting. Several open problems are also formulated.

Keywords: Membrane computing, P systems, recursively enumerable langauge, closure property

1 Introduction

Many classes of P systems with objects described by strings of symbols and processed by rewriting or by other operations were considered, see, e.g., [1], [2], [12], [5], [8], [9], [10], [11], [14] (an up-to-date bibliography of the area can be found at the web address http://bioinformatics.bio.disco.unimib.it/psystems). Recently, in [9] a variant was

[1]Work supported by a grant of NATO Science Committee, Spain, 2000–2001

considered, where rules of the form $X \to (u_1, tar_1)||\ldots||(u_n, tar_n)$ are used. When applying this rule to a string, n strings are obtained, by replacing one occurrence of the symbol X by $u_1, \ldots, u_n$ and replicating the remaining part of the string; the n strings are sent to the membranes indicated by the targets $tar_1, \ldots, tar_n$, respectively. Such systems were proven in [9] to characterize the recursively enumerable languages and to be able to solve SAT and HPP in linear time.

The characterization of RE languages from [9] is based on a proof which does not give a bound on the number of membranes; at the end of [9] one formulates the open problem whether or not the hierarchy on the number of membranes collapses.

We solve here this problem and we also strenghten the result from [9] from various points of view: six membranes suffice, even without using target indications of the form in_j (specifying the label of the destination), but only *in/out* commands (when it is sent *in*, a string goes to any directly lower membrane, nondeterministically chosen), without using a terminal alphabet for squeezing the accepted strings, and using only replication rules which produce at most two strings. We do not know whether or not this result is optimal, or a smaller number of membranes suffice. The power of systems with less than six membranes are compared with matrix languages and with E0L and ET0L languages, but without having a complete answer to the question where these latter families are placed in the hierarchy with respect to the number of membranes.

In the last section of the paper we shortly consider a "classic" problem in formal language theory which was never considered up to now in the P systems area: closure properties. We give some results about union, morphisms, and intersection with regular languages; the case of other AFL operations remain to be investigated.

2 Language Theory Prerequisites

In this section we introduce some formal language theory notions and notations which will be used in this paper; for further details we refer to [13].

For an alphabet V, by V^* we denote the set of all strings over V, including the empty one, denoted by λ. By *CF, RE* we denote the families of context-free and of recursively enumerable languages, respectively, while $E0L$ and $ET0L$ denote the families of languages generated by extended interactionless Lindenmayer (0L, for short) systems and by extended tabled 0L systems (ET0L systems). By pCF we denote the family of languages generated by *pure* context-free grammars, that is, context-free grammars with no distinction between terminals and nonterminals (all generated strings are accepted in the language). It is known that $pCF \subset CF \subset E0L \subset ET0L \subset RE$, all inclusions being proper.

In the proof from the next section we need the notion of a *matrix grammar with appearance checking*; such a grammar is a construct $G = (N, T, S, M, F)$, where N, 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 $N \cup T$, 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 (1) $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$, or (2) $w_i = w_{i+1}$, 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 – one says 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}. When $F = \emptyset$ (hence we do not use the appearance checking feature), the generated family is denoted by MAT.

It is known that $CF \subset MAT \subset MAT_{ac} = RE$, all inclusions being proper. All one-letter languages in the family MAT are regular, see [7].

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 $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$,
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 $x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3; $\#$ is called 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.

According to Lemma 1.3.7 in [4], for each matrix grammar there is an equivalent matrix grammar in the binary normal form.

For an arbitrary matrix grammar $G = (N, T, S, M, F)$, let us denote by $ac(G)$ the cardinality of the set $\{A \in N \mid A \to \alpha \in F\}$. From the construction in the proof of Lemma 1.3.7 in [4] one can see that if we start from a matrix grammar G and we get the grammar G' in the binary normal form, then $ac(G') = ac(G)$.

In [6] it is proved that each recursively enumerable language can be generated by a matrix grammar G such that $ac(G) \leq 2$. Consequently, to the properties of a grammar G in the binary normal form we can add the fact that $ac(G) \leq 2$. We will say that this is *the strong binary normal form* for matrix grammars.

3 Replicated Rewriting P Systems

We introduce here only the class of P systems with string-objects processed by replicated rewriting in the restricted form that we will investigate here (as usual, a membrane structure is represented by a string of labeled parentheses, and with each membrane we uniquely associate a *region*, that delimited by the membrane and the directly inner membranes, if any exists; we refer the regions by the labels of the corresponding membranes; for further basic elements of membrane computing area we refer to [12] or to other papers available about this subject).

An extended *replicated rewriting P system* (of degree $m \geq 1$) is a construct

$$\Pi = (V, T, \mu, M_1, \ldots, M_m, R_1, \ldots, R_m),$$

where:

1. V is the alphabet of the system;
2. $T \subseteq V$ is the *terminal alphabet*;
3. μ is a membrane structure with m membranes (in this section, we assume that the membranes are injectively labeled by $1, 2, \ldots, m$);
4. $M_1, \ldots, M_m$ are finite languages over V, representing the strings initially present in the regions $1, 2, \ldots, m$ of the system, respectively;
5. $R_1, \ldots, R_m$ are finite sets of *rules* of the form $X \to (u_1, tar_1)||\ldots||(u_n, tar_n)$, with $n \geq 1, X \in V, u_i \in V^*, tar_i \in \{here, out, in\}, 1 \leq i \leq n$, associated with the regions of μ.

A system is said to be *non-extended* if $V = T$.

The transitions in a system as above are defined as usual in rewriting P systems (in each region, each string which can be rewritten by a rule from that region is rewritten), with the following difference: when rewriting a string $x_1 X x_2$ by a rule $X \to (u_1, tar_1)||\ldots||(u_n, tar_n)$ we get n strings, $x_1 u_i x_2, 1 \leq i \leq n$, which will be communicated to the membranes indicated by $tar_i, 1 \leq i \leq n$, respectively (*here* means that the string remains in the same region, *out* means that the string exits from the region and goes to the surrounding region, and *in* means that the string is sent to one of the directly lower membranes, if any exists, otherwise the rule cannot be applied).

Thus, if $n = 1$, then we have a usual rewriting rule. When presenting a rewriting rule $X \to (u, tar)$, if the target is *here*, then it is omitted, and we simply write $X \to u$.

As usual, a sequence of transitions forms a computation and the result of a halting computation is the set of strings over T sent out of the system during the computation. In the case of non-extended systems, all strings sent out are accepted. A computation which never halts yields no result (this is very important in the proofs from the following sections, because a basic trick used in order to check the contents of a string is to replicate it, and to send one copy to a specific membrane, where certain rules just check whether or not we are "on a wrong path", a case where the computation never ends in that membrane). A string which remains inside the system or, in the extended case, exits but contains symbols not in T does not contribute to the generated language. The language generated in this way by a system Π is denoted by $L(\Pi)$.

The family of all languages $L(\Pi)$, computed as above by extended systems Π of degree at most $m \geq 1$ is denoted by $ERRP_m$; when using non-extended systems we get the family RRP_m. If we use only rewriting rules, without replication, then we denote by RP_m, ERP_m the corresponding families. If the degree of the systems is not bounded, then the subscript m is replaced by $*$.

In [9] it is proved that $RE = ERRP_*$ and one asks whether or not the hierarchy $ERRP_m, m \geq 1$, is infinite.

4 Computational Universality

We start by giving the main result of this paper, which solves the open problem from [9].

Theorem 4.1 $ERRP_6 = RRP_6 = RE$.

Proof. We only have to prove the inclusion $RE \subseteq RRP_6$. To this aim, we make use of the equality $RE = MAT_{ac}$.

Consider a matrix grammar with appearance checking, $G = (N, T, S, M, F)$, in the strong binary normal form, that is with $N = N_1 \cup N_2 \cup \{S, \#\}$, with rules of the four forms mentioned in Section 2, and with $ac(G) \leq 2$. Assume that we are in the worst case, with $ac(G) = 2$, and let $B^{(1)}, B^{(2)}$ be the two symbols in N_2 for which we have rules $B^{(j)} \to \#$ in matrices of M. Let us assume that we have k matrices of the form $m_i : (X \to \alpha, A \to x), X \in N_1, \alpha \in N_1 \cup \{\lambda\}, A \in N_2$, and $x \in (N_2 \cup T)^*$, $1 \leq i \leq k$ (that is, without rules to be used in the appearance checking manner). Each matrix of the form $(X \to \lambda, A \to x), X \in N_1, A \in N_2, x \in T^*$, is replaced by $(X \to f, A \to x)$, where f is a new symbol. We continue to label the obtained matrix in the same way as the original one. The matrices of the form $(X \to Y, B^{(j)} \to \#), X, Y \in N_1$ (that is, with rules used in the appearance checking manner), are labeled by m_i, with $i \in lab_j$, for $j = 1, 2$, such that lab_1, lab_2, and $lab_0 = \{1, 2, \ldots, k\}$ are mutually disjoint sets.

We construct the (non-extended) P system (of degree 6)

$$\Pi = (V, V, \mu, M_1, \ldots, M_6, R_1, \ldots, R_6),$$

with the following components:

$$\begin{aligned}
V &= T \cup N_1 \cup N_2 \cup \{(X_i, j), (X_i, j)' \mid X \in N_1, 1 \leq i \leq k, 0 \leq j \leq k\} \\
&\cup \{X_i \mid X \in N_1, i \in lab_1 \cup lab_2\} \\
&\cup \{(A_i, j), (A_i, j)' \mid A \in N_2, 1 \leq i \leq k, 0 \leq j \leq k\} \\
&\cup \{f, E, E', Z\}, \\
\mu &= [_1[_2[_3[_4\,]_4]_3]_2[_5\,]_5[_6\,]_6]_1, \\
M_1 &= \{XA\}, \text{ for } (S \to XA) \text{ being the initial matrix of } G, \\
M_i &= \emptyset, \text{ for all } i = 2, \ldots, 6,
\end{aligned}$$

and with the following sets of rules (the membranes with labels 2, 3, 4 will be used for simulating matrices $m_i, 1 \leq i \leq k$, while membranes with labels 5, 6 will simulate the matrices with labels in lab_1, lab_2, respectively):

R_1: $X \to (X_i, 0)$ and
$A \to ((A_i, 0), in)$, for $m_i : (X \to \alpha, A \to x), \alpha \in N_1 \cup \{f\}, 1 \leq i \leq k$,
$X \to (Y, here)||(X_i, in)$, for $m_i : (X \to Y, B^{(j)} \to \#), i \in lab_j, 1 \leq j \leq 2$,
$f \to (\lambda, out)||(f, in)$;

R_2: $X_i \to Z$, for all $X \in N_1, i \in lab_1 \cup lab_2$,
$(X_i, 0) \to ((X_i, 0), in)$, for all $X \in N_1, 1 \leq i \leq k$,
$E' \to (\lambda, out)$,
$A \to Z$, for all $A \in N_2$,
$Z \to Z$;

R_3: $(A_i, j) \to (A_i, j+1)'$, for all $A \in N_2, 0 \leq j < i \leq k$,
$(X_i, j) \to ((X_i, j+1)', in)$, for all $X \in N_1, 0 \leq j < i \leq k$,
$E \to (\lambda, out)$;

R_4: $(X_i, j)' \to (X_i, j)$, for $X \in N_1, 1 \leq j < i \leq k$,
$(A_i, j)' \to ((A_i, j), out)$, for $A \in N_2, 1 \leq j < i \leq k$,
$(X_i, i)' \to \alpha E'$, and
$(A_i, i)' \to (xE, out)$, for $m_i : (X \to \alpha, A \to x), 1 \leq i \leq k, \alpha \in N_1 \cup \{f\}$,

$R_{4+j,1}$: $(A_i, 0) \to Z$, for all $A \in N_2, i \in lab_0$,
$X_i \to Z$, for all $X \in N_1, i \notin lab_j$,
$B^{(j)} \to Z$,
$f \to Z$,
$Z \to Z$, for $j = 1, 2$.

(As usual, the target indication *here* is omitted from rewriting rules.)

Let us examine the work of this system. Assume that in the skin membrane we have a string Xw; initially, we have here XA, for $(S \to XA)$ the start matrix of G.

If we use a rule of the form $A \to ((A_i, 0), in)$, for some $m_i : (U \to \alpha, A \to x)$ in G, $1 \leq i \leq k$, then the string is sent to one of membranes 2, 5, 6. If the string does not contain a symbol $(X_i, 0)$, for some $X \in N_1$ and $1 \leq i \leq k$, then it will remain forever in membrane 2, where only rules of the form $A \to Z, Z \to Z$ can be applied to it. If the string arrives in one of membranes 5, 6, then the trap-symbol Z is produced and the computation will never stop.

Thus, before using the rule $A \to ((A_i, 0), in)$ we have to use also a rule $X \to (X_j, 0)$, for some $1 \leq j \leq k$. Again, if the string arrives in one of membranes 5, 6, then the trap-symbol is introduced. Therefore, we get a string of the form $(X_j, 0)w_1(A_i, 0)w_2$ which is sent to membrane 2, and from here to membrane 3, unchanged. In membrane 3, both tuple symbols will be replaced by primed tuple symbols, with the second component increased by one. If we use only a rule $(X_j, t) \to ((X_j, t+1)', in), t \geq 0$, then we sent to membrane 4 a string with only one primed symbol; after using the rule $(X_j, t+1)' \to (X_j, t+1)$ we cannot exit, hence we get no output. Therefore, in membrane 3 we have to use both rules $(A_i, 0) \to (A_i, 1)'$ and $(X_j, 0) \to ((X_j, 1)', in)$. This means that in membrane 4 we use both rules $(X_j, 1)' \to (X_j, 1)$ and $(A_i, 1)' \to ((A_i, 1), out)$ (if only the second rule is used, then the string will remain forever in membrane 3).

We return to membrane 3 the string $(X_j, 1)w_1(A_i, 1)w_2$. The process is iterated: we increase once again the second components of the tuple symbols, we pass to membrane 4 where we remove the primes, and so on and so forth. Always, both in membrane 3 and in

membrane 4 we have to rewrite both tuple symbols, otherwise the string gets stuck in one of these membranes.

We now distinguish three cases:

Case 1: $i < j$. This means that at some moment in membrane 3 we produce the string $(X_j, i)'w_1(A_i, i)'w_2$, which is sent to membrane 4. Here we have to use the rule $(A_i, i)' \to (xE, out)$ and we arrive in membrane 3 with the string $(X_j, i)'w_1xEw_2$, or with $(X_j, i)w_1xEw_2$. Each of these strings can be sent to membrane 2 by using the rule $E \to (\lambda, out)$, but it remains in membrane 2 forever and at most rules $A \to Z$ will be applied to it. The second string can be rewritten in membrane 3 also by the rule $(X_j, i) \to ((X_j, i+1)', in)$, hence the string $(X_j, i+1)'w_1xEw_2$ arrives in membrane 4. If $j \neq i+1$, then the prime is removed and the string remains forever in this membrane. If $j = i+1$, then we can use the rule $(X_{i+1}, i+1)' \to \alpha E'$ and again the string remains forever in this membrane. So, no output is obtained in this case.

Case 2: $i > j$. At some moment, in membrane 3 we produce the string $(X_j, j)'w_1(A_i, j)'w_2$, which is sent to membrane 4. If we use the rule $(A_i, j)' \to ((A_i, j), out)$ and we send the string $(X_j, j)'w_1(A_i, j)w_2$ to membrane 3, then it will remain forever here, in the form $(X_j, j)'w_1(A_i, j+1)w_2$, because no rule can be applied to it. If we first use here the rule $(X_j, j)' \to \alpha E'$, and then $(A_i, j)' \to ((A_i, j), out)$, then we get the string $\alpha E'w_1(A_i, j)w_2$ in membrane 3, and again the string will remain forever here, in the form $\alpha E'w_1(A_i, j+1)'w_2$. Also in this case we get no output.

Case 3: $i = j$. After i increasings of the second components of the tuple symbols, in membrane 3 we get the string $(X_i, i)'w_1(A_i, i)'w_2$. We send it to membrane 4, where the only possibility is to use the rules $(X_i, i)' \to \alpha E'$ and $(A_i, i)' \to (xE, out)$. If only the latter rule is used, then the string $(X_i, i)'w_1xEw_1$ is sent to membrane 3; the only applicable rule is $E \to (\lambda, out)$, the string is sent to membrane 2, where it will remain forever. If both rules $(X_i, i)' \to \alpha E', (A_i, i)' \to (xE, out)$ are used, then in membrane 3 we get the string $\alpha E'w_1xEw_2$, which can be sent to the skin membrane by using the rule $E \to (\lambda, out)$ in membrane 3 and $E' \to (\lambda, out)$ in membrane 2. What we get is exactly the result of simulating the matrix $m_i : (X \to \alpha, A \to x)$.

Consequently, we correctly simulate the matrices $m_i, 1 \leq i \leq k$, if and only if we start in the skin membrane by using the rules $X \to (X_j, 0), A \to ((A_i, 0), in)$, for $i = j$ (and with X and A corresponding to the same matrix m_i). The process can be iterated.

Assume now that in the skin membrane we rewrite a string Xw by means of a rule of the form $X \to (Y, here)||(X_i, in)$, for $i \in lab_j$ for some $j = 1, 2$ (this means that we intend to simulate a matrix with a rule used in the appearance checking mode). The string Yw remains in the skin membrane and can be processed as above, the string X_iw is sent to one of membranes 2, 5, 6. In membrane 2 it immediately introduces the trap-symbol. The same happens in that of membranes 5, 6 which is different from $4+j$. If the string arrives in the correct place, that is, in membrane $4+j$, then the trap-symbol is introduced if and only if the string contains the symbol $B^{(j)}$. Therefore, the computation will halt if and only if the simulation of the matrix $m_i : (X \to Y, B^{(j)} \to \#), i \in lab_j, j = 1, 2$, is correct. Again, the process can be iterated.

At the moment when we simulate a matrix $m_i : (X \to f, A \to x)$, we return to the skin membrane a string of the form fz. If z is not terminal and a rule of the form $A \to ((A_i, 0), in)$ is used, then either the string will remain forever in membrane 2, or the trap-symbol will

be introduced in membranes 5, 6 by one of the rules $(A_i, 0) \to Z$ or by $f \to z$. If we do not use a rule $A \to ((A_i, 0), in)$, then in the skin membrane we have to use the rule $f \to (\lambda, out)||(f, in)$. The string z is sent out, the string fz is sent to one of the lower membranes. In membranes 5, 6 we immediately introduce the trap-symbol, hence the string should arrive in membrane 2. If any nonterminal symbol $A \in N_2$ is still present in z, then a rule $A \to Z$ can be used and again the computation will never stop. Otherwise, the string remains here, no rule can be applied to it, but we know that the string z is a terminal one.

In conclusion, $L(\Pi)$ contains exactly the terminal strings which can be generated by G, that is, it is equal to $L(G)$. This completes the proof. □

It is an *open problem* whether or not the previous result is optimal, or the number of used membranes can be decreased.

5 The Power of Systems with Less Than Six Membranes

First, let us observe that when the matrix grammar from which we start the construction in the proof of Theorem 4.1 is without appearance checking, then four membranes suffice, that is, we have:

Corollary 5.1 $MAT \subseteq RRP_4$.

Moreover, in this case we use the replication only when checking that the string which is sent out is composed of terminal symbols with respect to G; this can be done also by using a terminal alphabet in the P system, that is, we can write:

Corollary 5.2 $MAT \subseteq ERP_4$.

The following relations are direct consequences of the definitions:

Lemma 5.1 (i) $RRP_m \subseteq ERRP_m, m \geq 1$.
(ii) $ERRP_m \subseteq ERRP_{m+1}, RRP_m \subseteq RRP_{m+1}, m \geq 1$.
(iii) $ERP_m \subseteq ERRP_m, RP_m \subseteq RRP_m, m \geq 1$.
(iv) $RRP_1 = pCF$ *and* $ERRP_1 = CF$.

Also, it is easy to prove that $RP_2 - CF \neq \emptyset$.

Somewhat expected (if we have in mind the proof of Theorem 4.1), we can avoid the use of a terminal alphabet by using a supplementary membrane:

Lemma 5.2 $ERRP_m \subseteq RRP_{m+1}, m \geq 1$.

Proof. Consider a system $\Pi = (V, T, \mu, M_1, \ldots, M_m, R_1, \ldots, R_m), m \geq 1$. We construct a system Π' of degree $m+1$ in the following manner. A further membrane is added around μ, as the skin membrane of Π'; we label it with 0 and we take $M_0 = \emptyset$. All axioms w of Π are

placed in the same regions of μ, in the form Xw, where X is a new symbol, which is added to the total alphabet of Π'. The set R_0 consists of the following rules:

$$\{X \to (\lambda, out)||(\lambda, here),\ Z \to Z\} \cup \{a \to Z \mid a \in V - T\}.$$

The symbol Z is a trap-symbol, once introduced it can be rewritten forever. (Z is not necessarily a new symbol, if such a trap-symbol already exists in the system Π, then we can use it.) After sending out a string z generated by Π, it arrives in membrane 0 in the form Xz; z is sent out and a copy of it is preserved in the skin membrane of Π'. If any symbol from $V - T$ is still present in it, then the computation never stops. Consequently, for Π' we can take the language generated in the non-extended manner and it equals the language $L(\Pi)$. □

The inclusion in Corollary 5.1 is proper, because the following result holds:

Theorem 5.1 *$RRP_4 - MAT \neq \emptyset$.*

Proof. Let us consider the system

$$\begin{aligned}
\Pi &= (V, V, [_1[_2[_3[_4\ ]_4]_3]_2]_1, \emptyset, \{ca\}, \emptyset, \emptyset, R_1, R_2, R_3, R_4),\\
V &= \{a, b, c, c', Z\},\\
R_1 &= \{c' \to (\lambda, out),\ a \to Z,\ Z \to Z\},\\
R_2 &= \{a \to bb,\ a \to (bb, in)||(bb, out),\ c \to (c', out)\},\\
R_3 &= \{b \to a,\ b \to (a, in)||(a, out)\},\\
R_4 &= \{b \to Z,\ Z \to Z\}.
\end{aligned}$$

We start in membrane 2 with the string ca; assume that we have here a string ca^n for some $n \geq 1$. The occurrences of a are replaced by bb. If we use the rule $a \to (bb, in)||(bb, out)$ before replacing each occurrence of a with bb, then in the skin membrane we will introduce the trap-symbol Z and the computation never stops. Otherwise, the string b^{2n} is rewritten in membrane 3 into a^{2n} and sent back to membrane 2; the rewriting should be complete, because if any occurrence of b is still present, then the trap-symbol is introduced in membrane 4. The process can be iterated, hence the number of occurrences of a is repeatedly doubled. At any moment, in membrane 2 we can use the rule $c \to (c', out)$; the string is sent to the skin membrane and it can immediately exit (with the symbol c erased).

Consequently, we have

$$L(\Pi) \cap a^* = \{a^{2^n} \mid n \geq 1\}.$$

(The intersection selects the strings to which no rule $a \to bb$ is used during the last passing through membrane 2.)

The family MAT is closed under intersection with regular languages, hence $L(\Pi) \notin MAT$. □

A result similar to Theorem 5.1 also holds for the case of ET0L languages:

Theorem 5.2 *$RRP_4 - ET0L \neq \emptyset$.*

Proof. Consider the following system:

$$\begin{aligned}
\Pi &= (V, V, [_1[_2[_3[_4\ ]_4]_3]_2]_1, \emptyset, \{c\}, \emptyset, \emptyset, R_1, R_2, R_3, R_4),\\
V &= \{a, b, c, d, e, e', Z\},\\
R_1 &= \{c'' \to (\lambda, out),\ c \to Z,\ e' \to Z,\ Z \to Z\},\\
R_2 &= \{c \to cde,\ c \to c'de,\ e' \to e,\\
&\quad d \to (a, in)||(a, out),\ c' \to (c'', out)\},\\
R_3 &= \{e \to be',\ e \to (be', in)||(be', out)\},\\
R_4 &= \{e \to Z,\ Z \to Z\}.
\end{aligned}$$

The system works in a way similar to the system in the proof of Theorem 5.1. We start in membrane 2 with the string c and we produce a string $c'(de)^n$, for some $n \geq 1$. When using the rule $d \to (a, in)||(a, out)$ we must have used already the rule $c \to c'de$, otherwise the trap-symbol is produced in the skin membrane. In membrane 3, each occurrence of e introduces a copy of b and is replaced by e'. This replacement should be complete in the moment when using the rule $e \to (be', in)||(be', out)$, otherwise Z is introduced in membrane 4. In membrane 2, each e' is replaced by e, then the rule $d \to (a, in)||(a, out)$ is again used (in this way we check whether or not any copy of e' is still present; in the affirmative case the rule $e' \to Z$ is used in the skin membrane). In membrane 3 each e will again introduce a copy of b. The process can be iterated at most n times, the number of copies of the symbol d in the string. The string can be sent out by using the rules $c' \to (c'', out), c'' \to (\lambda, out)$, from membranes 2 and 1, respectively. Thus, we obtain

$$L(\Pi) \cap \{a, b, e\}^* = \{(ab^n e)^n \mid n \geq 1\},$$

(the intersection with $\{a, b, e\}^*$ ensures the fact that no symbol d is present in the string).

The family $ET0L$ is closed under erasing morphisms and $\{(ab^n)^n \mid n \geq 1\}$ is not an ET0L language (see [13], page 272 of volume 1), which implies that $L(\Pi) \notin ET0L$. □

This result suggests to look also for a result like that in Corollary 5.1 for the family $ET0L$. We were only able to show that each ET0L language can be generated by using five membranes (and also using the extended alphabet feature); the optimality of this result remains *open*.

Theorem 5.3 $ET0L \subset ERRP_5$.

Proof. The properness follows from the previous theorem, we have only to prove the inclusion.

Each language $L \in ET0L$ can be generated by an ET0L system with only two tables, $G = (V, T, w, P_1, P_2)$, [13]. Assume that $P_i = \{a_{i,j} \to x_{i,j} \mid 1 \leq j \leq n_i\}$, for some $n_i \geq 1, i = 1, 2$. (Remember that each table in an ET0L system is *complete*, a rule $a \to x$ exists in each table for each symbol $a \in V$.) We construct the following P system (of degree five):

$$\begin{aligned}
\Pi &= (V', T, [_1[_2[_3\ ]_3]_2[_4\ ]_4[_5\ ]_5]_1, \{Xw\}, \emptyset, \emptyset, \emptyset, \emptyset, R_1, R_2, R_3, R_4, R_5),\\
V' &= V \cup \{a', a'' \mid a \in V\} \cup \{X, X_1, X_2, Z\},
\end{aligned}$$

$$
\begin{aligned}
R_1 &= \{X \to (\lambda, out)\} \\
&\cup \{a \to a',\ a \to a'' \mid a \in V\} \\
&\cup \{X \to (X, in)||(X_1, in),\ X \to (X, in)||(X_2, in)\}, \\
R_2 &= \{X_1 \to Z,\ X_2 \to Z,\ Z \to Z,\ X \to (X, in)||(X, out)\} \\
&\cup \{a'_{1,j} \to x_{1,j} \mid 1 \leq j \leq n_1\} \\
&\cup \{a''_{2,j} \to x_{2,j} \mid 1 \leq j \leq n_2\}, \\
R_3 &= \{a' \to Z,\ a'' \to Z \mid a \in V\} \\
&\cup \{Z \to Z\}, \\
R_4 &= \{X \to Z,\ X_2 \to Z,\ Z \to Z\} \\
&\cup \{a \to Z,\ a'' \to Z \mid a \in V\}, \\
R_5 &= \{X \to Z,\ X_1 \to Z,\ Z \to Z\} \\
&\cup \{a \to Z,\ a' \to Z \mid a \in V\}.
\end{aligned}
$$

This system works as follows. We start by having the axiom string of G together with the auxiliary symbol X in the skin membrane. Assume that we have here a string Xz, for arbitrary $X \in N_1$ and $z \in V^*$. By using the rules from R_1, some symbols are primed or double primed; eventually we also use one of the rules $X \to (X, in)||(X_1, in)$, $X \to (X, in)||(X_2, in)$, the string is duplicated and the copies are sent to lower membranes. The copy which contains the symbol X should arrive in membrane 2, otherwise the trap-symbol Z is introduced in membranes 4, 5, while the copy which contains one of the symbols X_1, X_2 cannot arrive in membrane 2, where it is replaced by Z. Moreover, if we have applied the first rule, that is, the string contains the symbol X_1, then it must arrive in membrane 4 (in membrane 5 this symbol is replaced by Z) and if we have applied the second rule, that is, the string contains the symbol X_2, then it must arrive in membrane 5. In this way we check two facts: that *all* symbols a from z were replaced by a' or by a'', and, moreover, that we have either introduced only single primed symbols or only double primed symbols, but not both of them. In this way, in membrane 2 we will correctly simulate either table 1 or table 2, without mixing the rules of the two tables.

In membrane 2, when simulating the rules of the two tables, we return to non-primed symbols. When using the rule $X \to (X, out)||(X, in)$ in membrane 2, a copy of the string is sent to the skin membrane and one is sent to membrane 3. In this latter membrane we check whether or not all primed or double primed symbol were replaced, that is, whether or not the simulation of the tables is done in the parallel manner specific to L systems (all symbols are rewritten in each step, and this is possible because the tables are complete). Only in the affirmative case we continue without introducing the trap-symbol.

We return to the skin membrane with a string similar to that we have started with. The process can be iterated. At any moment, we can use the rule $X \to (\lambda, out)$ and the string is sent out. If it is terminal, then it is accepted in the language of Π, if not, then it is "lost". Consequently, $L(\Pi) = L(G)$. □

As expected, the E0L languages can be generated by P systems with a smaller number of membranes.

Theorem 5.4 $E0L \subset ERRP_4$.

Proof. The strictness follows from Theorem 5.2, so we have to prove only the inclusion. Let $G = (V, T, w, P)$ be an E0L system. Let h be the morphism defined by $h(a) = a', a \in V$, where a' is a new symbol associated with a. We construct the P system (of degree four):

$$\begin{aligned}
\Pi &= (V', T, [_1[_2[_3[_4\ ]_4]_3]_2]_1, \emptyset, \{cw\}, \emptyset, \emptyset, R_1, R_2, R_3, R_4), \\
V' &= V \cup \{a' \mid a \in V\} \cup \{c, c', Z\}, \\
R_1 &= \{c' \to (\lambda, out),\ Z \to Z\} \\
&\cup \{a \to Z \mid a \in V\}, \\
R_2 &= \{a \to h(x),\ a \to (h(x), in)||(h(x), out) \mid a \to x \in P\} \\
&\cup \{c \to (c', out)\}, \\
R_3 &= \{a' \to a,\ a' \to (a, out)||(a, in) \mid a \in V\}, \\
R_4 &= \{a' \to Z,\ Z \to Z\}.
\end{aligned}$$

This system works in a way very similar to the system in the proof of Theorem 5.1, so we leave the details to the reader and we only mention that $L(G) = L(\Pi)$. □

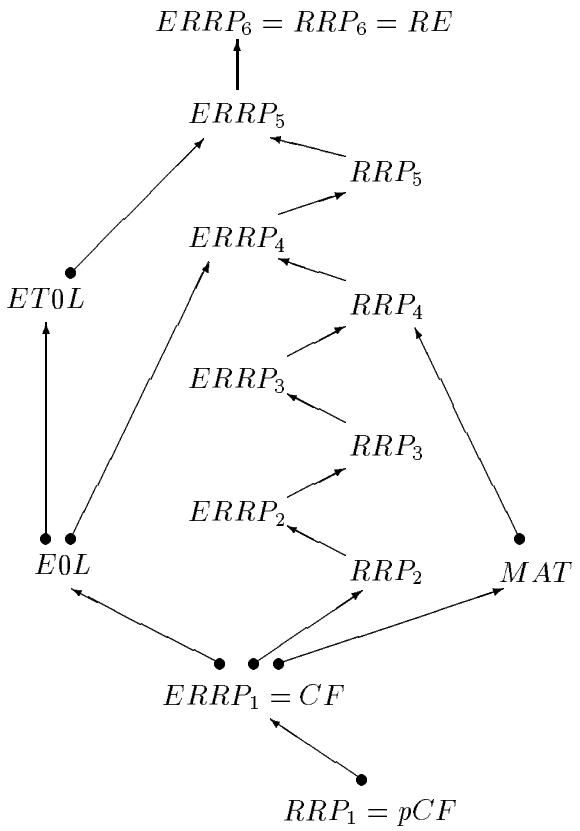

Figure 1

For an easy reading, we synthesize the relations from the previous theorems in the diagram from Figure 1; the arrows marked with a bullet correspond to proper inclusions, while the non-marked arrows indicate inclusions which are not known to be proper.

6 Closure Properties

One of the most investigated language-theoretic properties of families of languages is the closure with respect to certain operations, so this is a natural question also for language families generated by various classes of P systems. Of course, when a class of P systems characterizes RE, there is nothing to investigate, because the closure properties of RE are well-known, but when we get a smaller family (or a family which is not known to be equal to RE) the problem makes sense. However, up to now no paper has addressed this question for P families. Maybe, one of the reasons is that it seems to be a difficult questions – as we will also see below (in this moment, we do not have enough tools, such as normal forms and counterexamples, for approaching this problem).

The next theorem collects some closure properties about families $ERRP_m, m \geq 1$. Actually, these are not closure properties in the usual sense from formal language theory, but a sort of "growing (or absorbing) properties" of the hierarchy $ERRP_m, m \geq 1$: taking languages from a given family $ERRP_m$, the result of an operation on them is proven to belong to an upper family $ERRP_t$, for a well determined $t > m$ (of course, in view of Theorem 4.1, such a result is non-trivial only for $t \leq 5$). On the one hand, considering such a property for various increasing (finite or infinite) hierarchies of languages seems to be an interesting general language-theoretic problem, on the other hand, finding usual closure (or non-closure) properties for families in the P area remains as an *open problem.*

Theorem 6.1 (i) *If* $L_1 \in ERRP_{m_1}, L_2 \in ERRP_{m_2}$, *then* $L_1 \cup L_2 \in ERRP_{m_1+m_2+1}$. *If* Π_1, Π_2 *are two P systems with identical membrane structures of degree* m, *then* $L(\Pi_1) \cup L(\Pi_2) \in ERRP_{m+1}$.

(ii) *If* h *is a morphism,* R *is a regular language, and* $L \in ERRP_m, m \geq 1$, *then* $h(L) \in ERRP_{m+1}$ *and* $L \cap R \in ERRP_{m+1}$.

Proof (sketch). (i) It is enough to consider the case when the two languages are infinite, the case when one of them is finite being obvious (in such a situation, we do not have to increase the number of membranes necessary in order to generate the infinite language: introduce one further axiom in its skin, a symbol X, which just sends out the strings of the finite language).

In the general case, we assume the non-terminal symbols of each of Π_1, Π_2 different from the terminal symbols of the other system (a simple renaming can ensure this), then we just "put together" Π_1 and Π_2 in the same new skin membrane, where we also place the rules $a \to (a, out)$, for all symbols from the alphabets of Π_1, Π_2. The two systems will work independently and will send the computed strings out of their old skin membranes into the skin membrane of the new system; from here the strings are simply sent out. If any non-terminal symbol is present, then the string is "lost".

When the two systems have identical membrane structures we proceed as follows. Let μ be the common membrane structure, let V_1, V_2 be the total alphabets of the two systems, and

T_1, T_2 their terminal alphabets. As above, we assume that $(V_1 - T_1) \cap T_2 = \emptyset, (V_2 - T_2) \cap T_1 = \emptyset$. We construct a system Π with the terminal alphabet $T_1 \cup T_2$ and the membrane structure $\mu' = [_0 \mu]_0$ (we enclose μ in a new membrane, which is the skin membrane of the system Π). All axioms of Π_1 are introduced in the corresponding regions of μ with the symbols primed, and all the axioms of Π_2 are introduced in their regions with the symbols double primed. The same is done with the rules of the two systems: the rules of Π_1 are introduced with all symbols primed and the rules of Π_2 are introduced with the symbols duble primed, in the same regions as in the initial systems. In the skin membrane of Π we introduce the rules

$$a' \to a, \ a' \to (a, out), \ a \in V_1,$$
$$a'' \to a, \ a'' \to (a, out), \ a \in V_2.$$

It is easy to see that the work of the two systems Π_1, Π_2 is done in Π by the corresponding rules, without any interference. From the skin membrane of Π we send out the strings, unprimed (if any primed symbol remains in a string, then the string is not terminal). The equality $L(\Pi_1) \cup L(\Pi_2) = L(\Pi)$ is obvious.

(ii) Take a P system $\Pi = (V, T, \mu, M_1, \ldots, M_m, R_1, \ldots, R_m)$ and a deterministic finite automaton $A = (K, T, s_0, F, \delta)$. Construct the system $\Pi' = (K \times V \times K, \{(s, a, s') \mid s, s \in K, a \in T, s' = \delta(s, a)\}, \mu, M'_1, \ldots, M'_m, R'_1, \ldots, R'_m)$, where, for all $i = 1, 2, \ldots, m$, we have

$$\begin{aligned} M'_i \ = \ & \{(s_0, a_1, s_1)(s_1, a_2, s_2) \ldots (s_{k-1}, a_k, s_k) \mid s_j \in K, a_j \in V, 1 \leq j \leq k, \\ & k \geq 1, s_k \in F, a_1 a_2 \ldots a_k \in M_i\}, \end{aligned}$$

and the sets R'_i contain the rules obtained by using the classic triple construction, starting from the rules in R_i, plus $(s, a, s') \to Z$ for all $s, s' \in K, a \in V$, as well as $Z \to Z$. If h is the morphism defined by

$$h((s, a, s')) = a, \text{ for all } s, s' \in K, a \in V, s' = \delta(s, a),$$

then we have $L(\Pi) \cap L(A) = h(L(\Pi'))$.

Consequently, it is sufficient to prove the assertion for morphisms, and this is rather easy: take a morphism $h : T^* \longrightarrow U^*$, take a system $\Pi = (V, T, \mu, M_1, \ldots, M_m, R_1, \ldots, R_m)$, add one more membrane around μ, as the skin membrane of a new system Π'; the terminal alphabet of Π' is U; all axioms and all rules of Π are present in the corresponding membranes, with all symbols primed. In the skin membrane of Π' we introduce the rules

$$a' \to h(a), \ a' \to (h(a), out), \text{ for all } a \in T.$$

The equality $L(\Pi') = h(L(\Pi))$ is obvious. □

The case of the other three AFL operations – concatenation, Kleene closure, and inverse morphisms – remains to be investigated.

7 Final Remarks

We have investigated the P systems with string-objects, using evolution rules which at the same time can rewrite and replicate the strings, as introduced in [9]. The problem formulated

at the end of [9], whether or not the recursively enumerable languages can be characterized by such systems with a bounded number of membranes is affirmatively solved: systems with six membranes are universal (we do not know whether this result is optimal, or a smaller number of membranes suffices). We have also briefly examined the power of systems with less than six membranes, comparing them with matrix grammars, E0L, and ET0L systems. These "small systems" need further investigations, in particular, with respect to the closure properties of the corresponding families of languages, a problem which we find especially interesting from a technical point of view.

References

[1] P. Bottoni, A. Labella, C. Martin-Vide, Gh. Păun, Rewriting P systems with conditional communication, submitted, 2000.

[2] C. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor and Francis, London, 2000.

[3] J. Castellanos, A. Rodriguez-Paton, Gh. Păun, Computing with membranes: P systems with worm-objects, *IEEE 7th. Intern. Conf. on String Processing and Information Retrieval, SPIRE 2000*, La Coruna, Spain, 64–74.

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

[5] R. Freund, C. Martin-Vide, Gh. Păun, Computing with membranes: Three more collapsing hierarchies, submitted, 2000.

[6] R. Freund, Gh. Păun, On the number of non-terminals in graph-controlled, programmed, and matrix grammars, submitted, 2000.

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

[8] S. N. Krishna, R. Rama, On the power of P systems with sequential and parallel rewriting, *Intern. J. Computer Math.*, 77, 1-2 (2000), 1–14.

[9] S. N. Krishna, R. Rama, P systems with replicated rewriting, *J. Automata, Languages, and Combinatorics*, to appear.

[10] C. Martin-Vide, Gh. Păun, String objects in P systems, *Proc. of Algebraic Systems, Formal Languages and Computations Workshop*, Kyoto, 2000, RIMS Kokyuroku, Kyoto Univ., 2000.

[11] C. Martin-Vide, Gh. Păun, Computing with membranes. One more collapsing hierarchy, *Bulletin of the EATCS*, 72 (2000), 183–187.

[12] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143 (see also *Turku Center for Computer Science-TUCS Report* No 208, 1998, www.tucs.fi).

[13] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, Springer-Verlag, Heidelberg, 1997.

[14] Cl. Zandron, G. Mauri, Cl. Ferretti, Universality and normal forms on membrane systems, *Proc. Intern. Workshop Grammar Systems 2000* (R. Freund, A. Kelemenova, eds.), Bad Ischl, Austria, July 2000, 61–74.