

Reducing the Size of Extended Gemmating P Systems*

Erzsébet CSUHAJ-VARJÚ, György VASZIL

Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende utca 13-17, 1111 Budapest, Hungary
{csuhaj,vaszil}@sztaki.hu

Abstract

We improve results concerning the size bound of P systems with gemmation of mobile membranes. We show that systems with meta-priority relations and without (*in/out*) communication rules generate any recursively enumerable language with three membranes, thus we present an optimal result on the necessary number of membranes to obtain computational completeness. In the case of gemmating P systems with only pre-dynamical rules, we prove that four membranes are sufficient to generate any recursively enumerable language.

1 Introduction

Membrane systems, or P systems were introduced in [5] as distributed parallel computing devices of a biochemical type. The model is inspired by the functioning of the living cell: it consists of a membrane structure composed of several cell-membranes, hierarchically embedded in a membrane called the skin membrane. The membranes delimit regions and contain objects which evolve according to given evolution rules associated to the regions. Computations of the system are performed by the parallel and non-deterministic evolution of the contents of the membranes.

For a detailed introduction to the area of P systems, the interested reader is referred to the monograph [6]. An up-to-date bibliography of the field with an amount of additional information can be found at the web address <http://psystems.disco.unimib.it>.

In this paper we consider P systems with gemmation of mobile membranes or gemmating P systems, introduced in [3], which represent membrane systems with a new kind of communication.

The biological background of this new model can be summarized as follows: The cellular membranes are selectively permeable to small substances as, for example, water and gases, but not to bigger substances as proteins. These bigger substances are communicated among the cells by means of vesicles, encased on their cytosolic face by a specific protein which causes their budding from the membrane. When the

*Research supported in part by the Hungarian Scientific Research Fund "OTKA" Grants No. T 042529 and F 037567 and project "MolCoNet" IST-2201-32008.

vesicle fuses with its target membrane, the carried proteins are introduced inside it, where they can undergo different chemical reactions. This process can be modelled by so-called mobile membranes, that is, we can consider some objects in the original membrane to be transported by means of small membranes to a target membrane.

Gemmating P systems are with simple membrane structures, where the skin membrane contains only elementary membranes with string objects which correspond to proteins or any other structured bigger substances. These strings evolve according to operations with biochemical motivations, namely mutation, replication and splitting. The mutation in this case corresponds to the application of a context-free rule. Any membrane is provided with a set of classical evolution rules and a set of so-called pre-dynamical rules, which are rules defining the gemmation of the mobile membranes. There is a meta-priority relation defined between the set of classical evolution rules and the set of pre-dynamical rules which is needed to simulate the completion of the maturation path of an object. A pre-dynamical rule is a particular variant of an evolution rule which also indicates the membrane where the string must be communicated. After a pre-dynamical rule is used, the modified string object is transported into the target membrane, and from then it will evolve according to the rules of this membrane. This procedure corresponds to the gemmation and the fusion of the mobile membrane. In particular, the output of the system is due to the fusion of a mobile membrane with the skin membrane: this process causes the release of the objects outside the system and simulates the biological process of exocytosis.

In [2, 3] the computational power of P systems with gemmation of mobile membranes was examined. It was shown that these systems are as powerful as the Turing machines, in the case of extended systems (where a terminal set of objects is distinguished) with eight membranes [2] or with nine membranes if only pre-dynamical rules are allowed. These bounds were improved to five and six, respectively, in [1].

We continue the study of this area and show that extended systems with meta-priority relations but no (*in/out*) communication rules generate any recursively enumerable language with at most three membranes. Since these P systems with two membranes determine the class of context-free languages, the obtained result gives an optimal size bound for these variants of membrane systems to obtain computational completeness. We also prove that gemmating P systems with only pre-dynamical rules and with four membranes generate any recursively enumerable language. These variants of P systems with two membranes determine the class of finite languages, while having three membranes they define a language class which properly contains the class of regular languages. The exact place of the language class of gemmating P systems with three membranes and with only pre-dynamical rules is still open.

2 Preliminaries and Definitions

We first recall the notions and the notations we use. For more on the basic notions of formal language theory refer to [7]. Let V be an alphabet, let V^* be the set of all words over V , and let $V^+ = V^* - \{\varepsilon\}$ where ε denotes the empty word. The mirror image, or reverse of a word $x \in V^*$ is denoted by x^R , the set of natural numbers is

denoted by \mathbb{N} . We denote by FIN , REG , CF , and RE the class of finite, regular, context-free, and recursively enumerable languages, respectively. RE is the class accepted by Turing machines or generated by phrase structure grammars.

Now we present the notion of an Extended Post Correspondence (or EPC in short) from [4] and then define a set of context-free rewriting rules which can be derived from any such EPC and which will be used in the proofs of the results.

Definition 1 Let $\Sigma = \{a_1, \dots, a_n\}$, $1 \leq n$, be an alphabet. An *Extended Post Correspondence* (an EPC) is a pair

$$E = (\{(u_1, v_1), \dots, (u_m, v_m)\}, (z_{a_1}, \dots, z_{a_n})),$$

where $u_i, v_i, z_{a_j} \in \{0, 1\}^*$, $1 \leq i \leq m$, $1 \leq j \leq n$. The *language represented by E* , denoted by $L(E)$ is the following:

$$L(E) = \{x_1 \dots x_r \in \Sigma^* \mid \text{there are } i_1, \dots, i_s \in \{1, \dots, m\}, s \geq 1, \\ \text{such that } u_{i_1} \dots u_{i_s} z_{x_1} \dots z_{x_r} = v_{i_1} \dots v_{i_s}\}.$$

It is known (see [4]) that for each recursively enumerable language L there exists an EPC E such that $L(E) = L$.

Definition 2 Let E be an EPC as above, and let g, h be two mappings $g, h : \{0, 1\} \rightarrow \{A, B, C\}^*$ defined as $g(0) = AB$, $h(0) = C$, $g(1) = A$, $h(1) = BC$. Let us define a set P_E of rewriting rules containing the productions

$$\begin{aligned} S &\rightarrow g(u_i)Sh(v_i^R), \quad 1 \leq i \leq m, \\ S &\rightarrow S', \\ S' &\rightarrow g(z_a)S'a, \quad a \in \Sigma, \\ S' &\rightarrow \varepsilon \end{aligned}$$

constructed according to the EPC E .

Let us now observe the sentential forms that these rules generate. Starting from the symbol S , the rules of P_E generate strings of the form

$$g(u_{i_1} \dots u_{i_s} z_{x_1} \dots z_{x_r}) x_r \dots x_1 h((v_{i_1} \dots v_{i_s})^R)$$

where if $u_{i_1} \dots u_{i_s} z_{x_1} \dots z_{x_r} = v_{i_1} \dots v_{i_s}$, then $x_1 \dots x_r \in L(E)$, as above.

Let $u = u_{i_1} \dots u_{i_s}$, $v = v_{i_1} \dots v_{i_s}$, and $z_w = z_{x_1} \dots z_{x_r}$, for some $w = x_1 \dots x_r \in L(E)$. If $uz_w = v$, then the following properties must hold: The string $g(uz_w)$ either starts with AB or with AA , and $h(v^R)$ either ends with BC or with CC . Furthermore, if $g(uz_w)$ starts with AB , then $h(v^R)$ ends with CC , if $g(uz_w)$ starts with AA , then $h(v^R)$ ends with BC .

Based on these properties we might use the following procedure to produce any recursively enumerable language $L = L(E)$ for some EPC E .

Definition 3 Take an EPC E with $L = L(E)$, and construct the rule set P_E as above. The procedure $PROC_E$ consists of the instructions as follows.

Starting with the symbol S , use the rules of P_E to generate a string of the form $\alpha = g(uz_w) w^R h(v^R)$, u, z_w, v as above. Execute the following instructions on α .

1. Obtain α' by cutting an A from the left end of α .
2. Obtain α'' by cutting a C from the right end of α' .
3. Obtain α''' by cutting a B either from the left end or from the right end of α'' .
4. If $\alpha''' = x_r \dots x_1$, the reverse of $w = x_1 \dots x_r$, then $w \in L$, otherwise start again with executing the first instruction on α''' .

If at some point cutting the symbols from the ends of the string is not possible in the order required by the instructions above, then we might stop the procedure without any explicit conclusion. However, it is ensured that for each $w \in L$ there exists at least one α of the form above which leads us to the result of $w \in L$, thus, the procedure above produces all words belonging to $L(E)$.

Now we present the basic notions of membrane computing and then also define P systems with gemmation of mobile membranes. The interested reader may find more detailed information on the theory of P systems in the monograph [6].

A multiset is a pair $M = (V, f)$, where V is an arbitrary (not necessarily finite) set of objects and $f : V \rightarrow \mathbb{N}$ is a mapping which assigns to each object its multiplicity. The support of $M = (V, f)$ is the set $supp(M) = \{a \in V \mid f(a) \geq 1\}$. If V is a finite set, then M is called a finite multiset. The set of all finite multisets over the set V is denoted by V° .

We say that $a \in M = (V, f)$ if $a \in supp(M)$, and $M_1 = (V_1, f_1) \subseteq M_2 = (V_2, f_2)$ if $supp(M_1) \subseteq supp(M_2)$ and for all $a \in V_1$, $f_1(a) \leq f_2(a)$. The union of two multisets is defined as $(M_1 \cup M_2) = (V_1 \cup V_2, f')$ where for all $a \in V_1 \cup V_2$, $f'(a) = f_1(a) + f_2(a)$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2) = (V_1 - V_2, f'')$ where $f''(a) = f_1(a) - f_2(a)$ for all $a \in V_1 - V_2$, and the intersection of two multisets is $(M_1 \cap M_2) = (V_1 \cap V_2, f''')$ where for $a \in V_1 \cap V_2$, $f'''(a) = \min(f_1(a), f_2(a))$, $\min(x, y)$ denoting the minimum of $x, y \in \mathbb{N}$. We say that M is empty, denoted by ϵ , if its support is empty, $supp(M) = \emptyset$. When giving the elements x_1, \dots, x_n of a multiset M , we use double brackets as $M = \{\{x_1, \dots, x_n\}\}$ to distinguish from the set usual notation.

A P system is a structure of hierarchically embedded membranes, each having a label and enclosing a region containing a multiset of objects and possibly other membranes. The out-most membrane which is unique and it is called the skin membrane. The membrane structure is denoted by a sequence of matching parentheses where the matching pairs have the same label as the membranes they represent. If $x \in \{[i,]_i \mid 1 \leq i \leq n\}^*$ is such a string of matching parentheses of length $2n$, denoting a structure where membrane i contains membrane j , then $x = x_1 [i x_2 [j x_3]_j x_4]_i x_5$ for some $x_k \in \{[l,]_l \mid 1 \leq l \leq n, l \neq i, j\}^*$, $1 \leq k \leq 5$. If membrane i contains membrane j , and there is no other membrane, k , such that k contains j and i contains k (x_2 and x_4 above are strings of matching parentheses themselves), then we say that membrane i is the parent membrane of j , denoted by $i = parent(j)$, and at the same time, membrane j is one of the child membranes of i .

By the contents of a region we mean the multiset of objects which is contained by the corresponding membrane excluding those objects which are contained by any of its child membranes.

The evolution of the contents of the regions of a P system is described by rules associated to the regions. Applying the rules synchronously in each region, the system performs a computation by passing from one configuration to another one.

By [2], gemmating P systems use only membrane structures of depth 2. The skin membrane will be always labelled with the number 0, while the inner membranes will be labelled with the numbers $1, \dots, n$.

Gemmating P systems are defined with three types of rules with biochemical inspiration: mutation, replication, and splitting of a string. In this paper we use only mutation rules without (*in/out*) communication which are context free rules of the form $a \rightarrow u$, where $a \in V$ and $u \in V^*$.

With each region $i = 0, 1, \dots, n$ we associate two distinct sets of rules:

- A set C_i of classical evolution rules, that is, a set of mutation rules of the above form, and
- a set D_i of pre-dynamical evolution rules, that is, a set of mutation rules of the form $a \rightarrow u$, with $a \in V$, such that given a string $w_1 = w'_1 a$ (or $w_1 = a w'_1$) we obtain $w_2 = w'_1 u$ ($w_2 = u w'_1$, respectively), where $u = u' @_j$ ($u = @_j u'$, respectively) with $w'_1, u'_1 \in V^*$. The letter $@_j$ is a special symbol not in V and $j \in \{0, 1, \dots, n\}, j \neq i$.

Pre-dynamical rules may introduce the special symbol $@_j$ only at the ends of the string. We will always consider the set D_0 as an empty set, that is no pre-dynamical rule will ever be defined for the skin membrane. When a symbol $@_j$ appears in some string w present in a membrane i , for $j \neq i$, then inside the P system two sequential and dynamical communication processes take place. We say that a mobile membrane carries the string w from the originating membrane i to the target membrane j .

To keep the construction closer to the functioning of real cells, we define a meta-priority relation between the rules of set C_i and D_i , for all $i = 1, \dots, n$, by which all applicable classical rules in C_i must be used before any applicable pre-dynamical rule in D_i . We remark that we do not define any priority relation between rules in the set C_i neither between rules in the set D_i .

Definition 4 An extended P system Π with gemmation of mobile membranes of degree $n + 1$, $n \geq 0$, is

$$\Pi = (V, T, \mu, I_0, \dots, I_n, (C_0, D_0), (C_1, D_1), \dots, (C_n, D_n)),$$

where:

- V is an alphabet not containing the symbols $@_0, @_1, \dots, @_n$,
- $T \subseteq V$ is the output (terminal) alphabet,
- $\mu = [0 [1]_1 [2]_2 \dots [_{n-1}]_{n-1} [n]_n]_0$ is a membrane structure of depth 2 and degree $n + 1$,
- I_i , $1 \leq i \leq n$, are multisets of finite support over V^+ ,

- (C_i, D_i) , $0 \leq i \leq n$, are sets of classical evolution rules and sets of pre-dynamical evolution rules, respectively. The set C_i has a meta-priority over D_i as far as the application of all of its rules is concerned. The set D_0 is empty.

An extended gemmating P system works as follows: The regions are processed simultaneously, that is, in every step, inside each region, all the strings which can be the subject of an evolution rule are simultaneously rewritten. The rules to be applied can be nondeterministically chosen among all the applicable rules in accordance with the meta-priority defined over the set of classical rules and the set of pre-dynamical rules. This means that no pre-dynamical rule $d \in D_i$ can be applied if there exists at least one classical rule $c \in C_i$ which is applicable. At each step of a computation a string can be rewritten by one rule only. The strings resulting after the application of a rule can remain inside the membrane where they are placed, or can be communicated by mobile membranes to the regions specified by the target indications $@_i$, $0 \leq i \leq n$.

At any given moment, the membrane structure together with all multisets of objects associated with the regions defined by the membrane structure form the configuration of the system. However, since we do not consider splitting/recombination rules, the processing of each string is independent of the other strings present in the regions. This means that instead of simultaneously keeping track of every multiset of objects that can be found in the regions of the system, we might consider the strings traveling through the regions of the membrane structure independently of the contents of the region.

Definition 5 Let $[w]_i$ denote a string present in region i , $1 \leq i \leq n$. We say that

- $[w]_i$ directly derives $[w']_i$ by a classical evolution rule, denoted as $[w]_i \Rightarrow [w']_i$ if $w = w_1aw_2$, $w' = w_1uw_2$, and $a \rightarrow u \in C_i$, or
- $[w]_i$ directly derives $[w']_j$ by a pre-dynamical rule, denoted as $[w]_i \Rightarrow [w']_j$ if $w = a_1 \dots a_m$, $a_k \rightarrow u \notin C_i$ for any k , $1 \leq k \leq m$, and any $u \in V^*$, but $w = w_1a$ (or $w = aw_1$) and $w' = w_1u$ ($w' = uw_1$, respectively), for some $a \rightarrow u@_j \in D_i$ ($a \rightarrow @_ju \in D_i$, respectively).

A sequence of transitions forms a computation. A computation halts when there is no rule which can be applied to any string in the current configuration. The output of the P system Π (or the language of Π) is the set of strings over T expelled from the system during a halting computation, that is, the set of strings sent to region 0 during the computation. Non-halting computations provide no output.

Definition 6 Let Π be an extended gemmating P system of degree $n + 1$, for $n \geq 0$, as above. The language generated by Π is the set of strings

$$L(\Pi) = \{w' \in T^* \mid w \in I_i \text{ and } [w]_i \Rightarrow^* [w']_0 \text{ during a halting computation of } \Pi, 1 \leq i \leq n\},$$

where \Rightarrow^* denotes the reflexive and transitive closure of \Rightarrow .

Let $EGemP_n(MPri, n(in/out))$ and $EGemP_n(Dyn)$ denote the class of languages generated by extended gemmating P systems of degree n with meta-priority but without the use of (in/out) communication rules, and the class of languages generated by systems degree n with pre-dynamical rules only, respectively.

3 The Size of Extended Gemmating P Systems

We first show that extended gemmating P systems with meta-priority relations but no (in/out) communication rules generate any recursively enumerable language with at most three membranes. Since these P systems with two membranes determine the class of context-free languages, the obtained result gives an optimal size bound for these variants of membrane systems to obtain computational completeness.

Theorem 1 $EGemP_3(MPri, n(in/out)) = RE$.

Proof. Our proof is based on the procedure outlined at the end of the previous section. Let $L \subseteq \Sigma^*$, let E be an Extended Post Correspondence as above with $L^R = L(E)$, and let P_E be the set of rewriting rules based on the EPC E as described above.

Let $\Pi = (V, \Sigma, \mu, I_0, I_1, I_2, (\emptyset, \emptyset), (C_1, D_1), (C_2, D_2))$ where $I_0 = I_2 = \epsilon$, and

$$\begin{aligned} V &= \Sigma \cup \{S, S', A, B, C, C', C'', C''', T, T'\}, \\ \mu &= [0 [1 [1 [2]_2]_1]_0], \\ I_1 &= \{S\}, \\ C_1 &= P_E \cup \{C \rightarrow C', C'' \rightarrow C''', T \rightarrow T'\}, \\ D_1 &= \{A \rightarrow @_2 T, C''' \rightarrow @_2\}, \\ C_2 &= \{C' \rightarrow C'', T' \rightarrow \epsilon, C''' \rightarrow C\}, \\ D_2 &= \{B \rightarrow @_0, B \rightarrow @_1, T \rightarrow @_1 T'\}. \end{aligned}$$

This system works as follows. Starting with the start symbol S in I_1 , the rules of P_E and $C \rightarrow C'$ generate a string of the form $uzwv$, $uz \in \{A, B\}^*$, $w \in \Sigma^*$, $v \in \{B, C'\}^*$ where if uz and v can be deleted using the instructions of the procedure $PROC_E$ in the previous section (with cutting C' instead of C in step 2), then $w \in L$.

Now, using the pre-dynamical rules of D_1 , an A on the left end of $uzwv$ is changed to T , and the result is sent to region 2. Here the rules of C_2 rewrite every C' to C'' . The result is sent back to region 1 by the pre-dynamical rules of D_2 and at the same time either a B is cut off the right end of the string, or the T on the left end is changed to T' . (If $B \rightarrow @_0$ is used, then the computation produces no result, since the nonterminal T is still present in the string sent out of the system.) In the first case, when T is not changed to T' , the rules $T \rightarrow T$ of C_1 lead the computation to an infinite loop. In the second case, all C'' is rewritten to C''' and the result is sent back to region 2 after cutting off a C''' from the right end. Now T' is erased, all C''' is changed to C , and the result is sent back to region 1 by cutting off a B from the left or right end of the string, or sent out of the system by $B \rightarrow @_0$. If the string is sent out and it is terminal, then it is the reverse of some $w \in L^R = L(E)$,

thus it is an element of $(L^R)^R = L$. If it is not sent out, then the system continues in the same way by cutting symbols off the ends of the string as long as it either enters an infinite loop, or sends out a non-terminal string, or produces a result by sending out a terminal word.

These steps execute instructions of the procedure $PROC_E$ described in the previous section, so our statement is proved. \square

In order to demonstrate that the above result is optimal, we note that

$$EGemP_2(MPri, n(in/out)) = CF.$$

Certainly, in this case, the only membrane which is able to modify the string is membrane 1. Furthermore, a word can be sent out to membrane 0 only if no mutation (context-free) rule of membrane 1 can be applied to it. Then, it is easy to see that for any context-free grammar $G = (N, T, P, S)$, where N is the set of nonterminals, T is the set of terminals, P is the set of productions, and S is the start symbol, the language $L(G)$ of G can be generated by the extended gemmating P system Π , with meta-priority relations but no (in/out) communication rules and with two membranes, where membrane 1 is defined by $I_1 = \{S\}$, $C_1 = P$, and $D_1 = \{a \rightarrow @_0 \mid a \in T\}$. If the empty word is in $L(G)$, then we add the rule $S \rightarrow @_0$ to D_1 . The fact, that no more than a context-free language can be obtained by these systems is obvious.

Now we prove that gemmating P systems with only pre-dynamical rules and with four membranes generate any recursively enumerable language. These variants of P systems with two membranes determine the class of finite languages, while having three membranes they define a language class which properly contains the class of regular languages.

Theorem 2 $EGemP_4(Dyn) = RE$.

Proof. Let $L \subseteq \Sigma^*$ where $\Sigma = \{a_1, \dots, a_n\}$, let E be an Extended Post Correspondence as above with $L^R = L(E)$, and let P_E be the set of rewriting rules based on the EPC E as described above. Let the rules of P_E be denoted as

$$P_E = \{S \rightarrow \alpha_i S \beta_i, S \rightarrow S', S' \rightarrow \gamma_j S' a_j, S' \rightarrow \varepsilon \mid 1 \leq i \leq m, 1 \leq j \leq n\},$$

where $\alpha_i = g(u_i)$, $\beta_i = h(v_i^R)$, $\gamma_j = g(z_{a_j})$ for some pair (u_i, v_i) , $1 \leq i \leq m$, and z_{a_j} , $a_j \in \Sigma$, $1 \leq j \leq n$, of the EPC E with mappings g, h as above.

Let $\Pi = (V, \Sigma, \mu, I_0, I_1, I_2, I_3, D_0, D_1, D_2, D_3)$, where $I_0 = I_2 = I_3 = \epsilon$, $D_0 = \emptyset$, and

$$\begin{aligned} V &= \Sigma \cup \{S_1, S_2, S'_1, S'_2, \bar{S}_1, X, \bar{X}, A, B, C, \$\}, \\ \mu &= [0 \ [1 \]_1 \ [2 \]_2 \ [3 \]_3 \]_0, \\ I_1 &= \{S_1 \gamma_i a_i S_2 \mid 1 \leq i \leq n\} \\ &\cup I'_1 \text{ where } I'_1 = \{\{\$\}\} \text{ if } \varepsilon \in L, \text{ otherwise } I'_1 = \epsilon, \\ D_1 &= \{S'_1 \rightarrow @_2 \bar{S}_1 X^{2i} S'_1 \alpha_i, S'_2 \rightarrow \beta_i S'_2 \bar{X}^{2i} @_3, S_1 \rightarrow @_2 S'_1, \\ &\quad S_1 \rightarrow @_2 \bar{S}_1 X^{2m+2j} S_1 \gamma_j, S_2 \rightarrow a_j S_2 \bar{X}^{2m+2j} @_3, \end{aligned}$$

$$\begin{aligned}
& S'_1 \rightarrow @_2 \bar{S}_1, S'_2 \rightarrow @_2, B \rightarrow @_2, B \rightarrow @_0, S'_2 \rightarrow @_0 \\
& \$ \rightarrow @_0 \mid 1 \leq i \leq m, 1 \leq j \leq n\}, \\
D_2 &= \{\bar{S}_1 \rightarrow @_1, S_2 \rightarrow S'_2 @_3, \bar{X} \rightarrow @_3, A \rightarrow @_3\}, \\
D_3 &= \{X \rightarrow @_2, S_1 \rightarrow @_1 S_1, S'_1 \rightarrow @_1 S'_1, C \rightarrow @_1\}.
\end{aligned}$$

This system works by executing the procedure $PROC_E$ described in the previous section. First, a string of the form $uzwv$, $uz \in \{A, B\}^*$, $w \in \Sigma^+$, $v \in \{B, C\}^*$ is created where if uz and v can be deleted using the instructions of $PROC_E$, then $w \in L$. The empty word is treated separately, if $\varepsilon \in L$, then it is generated by the rule $\$ \rightarrow @_0$.

The work of the system starts in region 1 with words of the form $S_1 \gamma_i a_i S_2$, $1 \leq i \leq n$. Let $S_1 \gamma a S_2$ denote one of these strings and let us follow its possible route through the regions of Π . If the rule for S_2 is applied first, then there is no terminal word produced from this string, since

$$[S_1 \gamma a S_2]_1 \Rightarrow [S_1 \gamma a a_i S_2 \bar{X}^{2m+2i}]_3 \Rightarrow [S_1 \gamma a a_i S_2 \bar{X}^{2m+2i}]_1 \Rightarrow$$

1. $[\bar{S}_1 X^{2m+2j} S_1 \gamma_j \gamma a a_i S_2 \bar{X}^{2m+2i}]_2 \Rightarrow$
 - (a) $[X^{2m+2j} S_1 \gamma_j \gamma a a_i S_2 \bar{X}^{2m+2i}]_1$
 - (b) $[\bar{S}_1 X^{2m+2j} S_1 \gamma_j \gamma a a_i S_2 \bar{X}^{2m+2i-1}]_3$
2. $[S'_1 \gamma a a_i S_2 \bar{X}^{2m+2i}]_2 \Rightarrow [S'_1 \gamma a a_i S_2 \bar{X}^{2m+2i-1}]_3 \Rightarrow$
 $[S'_1 \gamma a a_i S_2 \bar{X}^{2m+2i-1}]_1 \Rightarrow [\bar{S}_1 x \gamma a a_i S_2 \bar{X}^{2m+2i-1}]_2 \Rightarrow$
 - (a) $[x \gamma a a_i S_2 \bar{X}^{2m+2i-1}]_1$
 - (b) $[\bar{S}_1 x \gamma a a_i S_2 \bar{X}^{2m+2i-2}]_3$ where $x = \varepsilon$ or $x = X^{2j} S'_1 \alpha_j$,

thus, one of the rules for S_1 has to be applied. (Note that γ either starts with an A or $\gamma = \varepsilon$, so no rule can be applied to the string of point 2.(a) in region 1. even if $x = \varepsilon$.) Let us assume first that the rule $S_1 \rightarrow @_2 \bar{S}_1 X^{2m+2i} S_1 \gamma_i$ is used.

$$[S_1 \gamma a S_2]_1 \Rightarrow [\bar{S}_1 X^{2m+2i} S_1 \gamma_i \gamma a S_2]_2 \Rightarrow$$

1. $[\bar{S}_1 X^{2m+2i} S_1 \gamma_i \gamma a S'_2]_3$
2. $[X^{2m+2i} S_1 \gamma_i \gamma a S_2]_1 \Rightarrow [X^{2m+2i} S_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2m+2j}]_3 \Rightarrow$
 $[X^{2m+2i-1} S_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2m+2j}]_2 \Rightarrow$
 $[X^{2m+2i-1} S_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2m+2j-1}]_3 \Rightarrow \dots \Rightarrow$
 - (a) $[X^{2i-2j} S_1 \gamma_i \gamma a a_j S_2]_3 \Rightarrow [X^{2i-2j-1} S_1 \gamma_i \gamma a a_j S_2]_2 \Rightarrow$
 $[X^{2i-2j-1} S_1 \gamma_i \gamma a a_j S'_2]_3 \Rightarrow [X^{2i-2j-2} S_1 \gamma_i \gamma a a_j S'_2]_2$
 - (b) $[S_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i}]_3 \Rightarrow [S_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i}]_1 \Rightarrow$
 - i. $[\bar{S}_1 X^{2k} S_1 \gamma_k \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i}]_2 \Rightarrow$
 - A. $[X^{2k} S_1 \gamma_k \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i}]_1$
 - B. $[\bar{S}_1 X^{2k} S_1 \gamma_k \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i-1}]_3$

$$\begin{aligned}
& \text{ii. } [S'_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i}]_2 \Rightarrow [S'_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i-1}]_3 \Rightarrow \\
& [S'_1 \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i-1}]_1 \Rightarrow [\bar{S}_1 x \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i-1}]_2 \Rightarrow \\
& \text{A. } [x \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i-1}]_1 \\
& \text{B. } [\bar{S}_1 x \gamma_i \gamma a a_j S_2 \bar{X}^{2j-2i-2}]_3 \text{ where } x = \varepsilon \text{ or } x = X^{2k} S'_1 \alpha_k, \\
& \text{(c) } [S_1 \gamma_i \gamma a a_j S_2]_3 \Rightarrow [S_1 \gamma_i \gamma a a_j S_2]_1 \quad (\text{if } i = j)
\end{aligned}$$

The system cannot produce any terminal word from the strings of the above cases except case 2.(c) where we have a word of the form $S_1 \gamma_i \gamma_j a_j a_i S_2$ for $S' \rightarrow \gamma_i S' a_i$ and $S' \rightarrow \gamma_j S' a_j$, rules of P_E associated to the EPC E , $1 \leq i, j \leq n$, $a_i, a_j \in \Sigma$. The system may continue by appending corresponding strings and terminal symbols to the left and right ends of the sentential form in the same way as above, producing strings of the form $S_1 \gamma_{i_1} \dots \gamma_{i_j} a_{i_j} \dots a_{i_1} S_2$, or it may choose to apply the rule $S_1 \rightarrow @_2 S'_1$, in which case we get

$$[S_1 \gamma w S_2]_1 \Rightarrow [S'_1 \gamma w S_2]_2 \Rightarrow [S'_1 \gamma w S'_2]_3 \Rightarrow [S'_1 \gamma w S'_2]_1$$

where $\gamma \in \{A, B\}^*$, $w \in \Sigma^+$.

We obtain a word of the same form if at the beginning of the work of the system, the rule $S_1 \rightarrow @_2 S'_1$ is applied on an initial sentential form $S_1 \gamma_i a_i S_2$ in region 1 as

$$[S_1 \gamma_i a_i S_2]_1 \Rightarrow [S'_1 \gamma_i a_i S_2]_2 \Rightarrow [S'_1 \gamma_i a_i S'_2]_3 \Rightarrow [S'_1 \gamma_i a_i S'_2]_1,$$

so we may continue by assuming that we have a string of the form $S'_1 \gamma w S'_2$ with γ, w as above in region 1.

If one of the rules for S'_2 are applied, then the derivation stops without producing a terminal word as

$$[S'_1 \gamma w S'_2]_1 \Rightarrow [S'_1 \gamma w]_2, \text{ or } [S'_1 \gamma w S'_2]_1 \Rightarrow [S'_1 \gamma w]_0,$$

or as

$$[S'_1 \gamma w S'_2]_1 \Rightarrow [S'_1 \gamma w \beta_i S'_2 \bar{X}^{2i}]_3 \Rightarrow [S'_1 \gamma w \beta_i S'_2 \bar{X}^{2i}]_1 \Rightarrow [\bar{S}_1 x \gamma w \beta_i S'_2 \bar{X}^{2i}]_2 \Rightarrow$$

1. $[x \gamma w \beta_i S'_2 \bar{X}^{2i}]_1$
2. $[\bar{S}_1 x \gamma w \beta_i S'_2 \bar{X}^{2i-1}]_3$ where $x = \varepsilon$, or $x = X^{2j} S'_1 \alpha_j$,

thus, one of the rules for S'_1 has to be applied. (Note that either the first symbol of γ is an A , or γ is empty, so even if $x = \varepsilon$, no rule can be applied to the sentential form of point 1. in region 1.)

If the rule $S'_1 \rightarrow @_2 \bar{S}_1$ is used we have the following possibilities.

$$[S'_1 \gamma w S'_2]_1 \Rightarrow [\bar{S}_1 \gamma w S'_2]_2 \Rightarrow [\gamma w S'_2]_1 \Rightarrow$$

1. $[\gamma w \beta_i S'_2 \bar{X}^{2i}]_3$
2. $[\gamma w]_2 \Rightarrow [\gamma' w]_3$ where $\gamma = A \gamma'$
3. $[\gamma w]_0$, so if $\gamma = \varepsilon$, then $w \in L$.

The strings of cases 1. and 2. cannot produce any terminal words, in case 3., if $\gamma = \varepsilon$, then w , the corresponding terminal word is correctly output by the system.

If the rule $S'_1 \rightarrow @_2 \bar{S}_1 X^{2i} S'_1 \alpha_i$ is applied, we have

$$\begin{aligned}
& [S'_1 \gamma w S'_2]_1 \Rightarrow [\bar{S}_1 X^{2i} S'_1 \alpha_i \gamma w S'_2]_2 \Rightarrow [X^{2i} S'_1 \alpha_i \gamma w S'_2]_1 \Rightarrow \\
& 1. [X^{2i} S'_1 \alpha_i \gamma w]_0 \\
& 2. [X^{2i} S'_1 \alpha_i \gamma w]_2 \\
& 3. [X^{2i} S'_1 \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j}]_3 \Rightarrow [X^{2i-1} S'_1 \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j}]_2 \Rightarrow \\
& \quad [X^{2i-1} S'_1 \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j-1}]_3 \Rightarrow \dots \Rightarrow \\
& \quad (a) [X^{2i-2j} S'_1 \alpha_i \gamma w \beta_j S'_2]_3 \Rightarrow [X^{2i-2j-1} S'_1 \alpha_i \gamma w \beta_j S'_2]_2 \\
& \quad (b) [S'_1 \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j-2i}]_3 \Rightarrow [S'_1 \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j-2i}]_1 \Rightarrow \\
& \quad \quad [\bar{S}_1 x \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j-2i}]_2 \Rightarrow \\
& \quad \quad \quad i. [x \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j-2i}]_1 \\
& \quad \quad \quad ii. [\bar{S}_1 x \alpha_i \gamma w \beta_j S'_2 \bar{X}^{2j-2i-1}]_3 \text{ where } x = \varepsilon, \text{ or } x = X^{2k} S'_1 \alpha_k \\
& \quad (c) [S'_1 \alpha_i \gamma w \beta_j S'_2]_3 \Rightarrow [S'_1 \alpha_i \gamma w \beta_j S'_2]_1 \quad (\text{if } i = j)
\end{aligned}$$

(Note that either the first symbol of $\alpha_i \gamma$ is an A , or $\alpha_i \gamma$ is empty, so even if $x = \varepsilon$, no rule can be applied to the sentential form of point 3.(b)i. in region 1.) Thus, the system produced a string of the form $S'_1 \alpha_i \gamma w \beta_j S'_2$ for some rule $S \rightarrow \alpha_i S \beta_i$, $1 \leq i \leq m$, of P_E associated to E . It may continue to add corresponding string pairs to the two ends of the string as above producing a string of the form $S'_1 \alpha \gamma w \beta S'_2$, $\alpha, \beta \in \{A, B\}^*$, $\beta \in \{B, C\}^*$, $w \in \Sigma^+$, or it may choose to apply the rule $S'_1 \rightarrow @_2 \bar{S}_1$.

$$[S'_1 \alpha \gamma w \beta S'_2]_1 \Rightarrow [\bar{S}_1 \alpha \gamma w \beta S'_2]_2 \Rightarrow [\alpha \gamma w \beta S'_2]_1 \Rightarrow$$

1. [$\alpha \gamma w \beta \beta_i S'_2 \bar{X}^{2i}$]₃
2. [$\alpha \gamma w \beta$]₀
3. [$\alpha \gamma w \beta$]₂

No terminal string is produced in case 1. In case 2., if $\alpha \gamma = \beta = \varepsilon$, then w is correctly output by the system. In case 3., the execution of the erasing instructions of $PROC_E$ may start. First an A is erased in region 2 and the string is sent to region 3, where a C is erased. These must have been on the left and right ends of the string, respectively. Now a B is erased in region 1 from one of the two ends of the string and it is either sent out of the system or the erasing process might continue in region 2.

From these considerations we might see that Π correctly simulates $PROC_E$, thus our statement is proved. \square

Before closing the section, we add some remarks about the question of the optimality of the above result. We can immediately see that gemmating P systems with only pre-dynamical rules and two membranes determine the class of finite languages, thus

$$EGemP_2(Dyn) = FIN.$$

Certainly, any finite language $L = \{w_1, \dots, w_n\}$, where $w_i \in T^*$, for some alphabet T and $1 \leq i \leq n$, can be obtained with a system where $I_1 = \{S\}$ and $D_1 = \{S \rightarrow w_i@_0 \mid 1 \leq i \leq n\}$. The fact that membrane 1 is able to send out only a finite number of words is obvious.

Moreover,

$$REG \subset EGemP_3(Dyn).$$

In this case, only membrane 1 and membrane 2 are able to modify the strings they have, by appending words to its left-end and/or to its right-end. The language of the system is determined by the interplay of these two membranes. Now, suppose that $G = (N, T, P, S)$, is a regular grammar with productions of the form $A \rightarrow aB$ and $A \rightarrow a$, where A, B are nonterminals and a is a terminal symbol. Then, the gemmating P system with $I_1 = \{S\}$, $I_2 = \emptyset$, $D_1 = \{A \rightarrow A'@_2\}$, and $D_2 = \{A' \rightarrow aB@_1 \mid A \rightarrow aB \in P\} \cup \{A' \rightarrow a@_0\}$ determines $L(G)$, the language of G . For the gemmating P system with $I_1 = \{AB\}$, $I_2 = \emptyset$, $D_1 = \{B \rightarrow bB@_2, B \rightarrow b@_2\}$, and $D_2 = \{A \rightarrow @_1Aa, A \rightarrow a@_0\}$, we obtain the language $L = \{a^n b^n \mid n \geq 1\}$, which is a linear non-regular language. It is an open question how large computational power can be obtained with gemmating P systems with only pre-dynamical rules and three membranes.

4 Conclusion

In this paper we showed that extended gemmating P systems with even a minimal configuration, with three membranes, are as powerful as Turing machines. Since such constructions with 2 membranes can determine only the context-free language class, the obtained result is an optimal result. The result is interesting, since the size of the distributed architectures in molecular computing usually represents a separator between the class of regular languages and the class of recursively enumerable languages. In addition to this statement, we also proved that gemmating P systems with only pre-dynamical rules and with four membranes are computationally complete tools as well. These constructs with two membranes determine the class of finite languages, and with three membranes they are more powerful than the regular grammars. However, the exact computational power of gemmating P systems with pre-dynamical rules and with three membranes is still open.

References

- [1] D. Besozzi, E. Csuha-j-Varjú, G. Mauri, C. Zandron, Size and power of extending gemmating P systems. In: Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini (eds.), *Proc. Second Brainstorming Week on Membrane Computing, Sevilla, 2-7 February 2004*, TR 01/2004, Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, 2004, 92–101.
- [2] D. Besozzi, G. Mauri, Gh. Păun, C. Zandron, Gemmating P systems: collapsing hierarchies. *Theoretical Computer Science*, 296 (2):253–267, 2003.

- [3] D. Besozzi, C. Zandron, G. Mauri, N. Sabadini, P systems with gemmation of mobile membranes. In A. Restivo, S. Ronchi Della Rocca, L. Roversi (eds.), *Proceedings of the 7th Italian Conference of Theoretical Computer Science 2001*, volume 2202 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2001, 136–153.
- [4] V. Geffert, Context-free-like forms for the phrase structure grammars. In: M. P. Chytil, L. Janiga, V. Koubek (eds.), *Mathematical Foundations of Computer Science 1988, 13th Symposium Carlsbad, Czechoslovakia, August 29 - September 2, 1988. Proceedings*, volume 324 of *Lecture Notes in Computer Science*, Springer-Verlag, 1988, 309–317.
- [5] 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 208*, 1998, www.tucs.fi).
- [6] Gh. Păun, *Computing with Membranes: An Introduction*, Springer-Verlag, Berlin, 2002.
- [7] G. Rozenberg, A. Salomaa (eds.) *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.