Fundamenta Informaticae XX (2009) 1–15
IOS Press

P Systems with Transport and Diffusion Membrane Channels

Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo
Dipartimento di Informatica, Università di Pisa
Largo Pontecorvo 3, 56127 Pisa, Italy
Email: {barbuti, maggiolo, milazzo}@di.unipi.it

Simone Tini
Dipartimento di Scienze della Cultura, Politiche e dell'Informazione
Università dell'Insubria
Via Carloni 78, 22100 Como, Italy
Email: simone.tini@uninsubria.it

Abstract. P Systems are computing devices inspired by the structure and the functioning of a living cell. A P System consists of a hierarchy of membranes, each of them containing a multiset of objects, a set of evolution rules, and possibly other membranes. Evolution rules are applied to the objects of the same membrane with maximal parallelism. In this paper we present an extension of P Systems, called P Systems with Membrane Channels (PMC Systems), in which membranes are enriched with channels and objects can pass through a membrane only if there are channels on the membrane that enable such a passage. We show that PMC Systems are universal even if only the simplest form of evolution rules is considered, and we give two application examples.

1. Introduction

P Systems were introduced by Păun in [4] as distributed parallel computing devices inspired by the structure and the functioning of a living cell. A P System consists of a *hierarchy of membranes*, each of them containing a multiset of *objects*, representing molecules, a set of *evolution rules*, representing chemical reactions, and possibly other membranes. For each evolution rule there are two multisets of objects, describing the reactants and the products of the chemical reaction. A rule in a membrane can be applied only to objects in the same membrane. Some objects produced by the rule remain in the same

Address for correspondence: Dipartimento di Informatica, Università di Pisa, Largo Pontecorvo 3, 56127 Pisa, Italy

membrane, others are sent *out* of the membrane, others are sent *into* the inner membranes, which are identified by their labels. Evolution rules are applied with *maximal parallelism*, meaning that it cannot happen that some evolution rule is not applied when the objects needed for its triggering are available.

Many variants and extensions of P Systems exist that include features to increase their expressiveness and that are based on different evolution strategies. Among the most common extensions we mention P Systems with dissolution rules that allow a membrane to disappear and release in the environment all the objects it contains. We mention also P Systems with priorities, in which a priority relationship exists among the evolution rules of each membrane and can influence the applicability of such rules, and P Systems with promoters and inhibitors, in which the applicability of evolution rules depends on the presence of at least one occurrence and on the absence, respectively, of a specific object. Moreover, we mention P Systems with symport/antiport rules that allow simultaneous trans-membrane transportation of objects either in the same direction (symport) or in opposite directions (antiport). See [1, 5] for the definition of these (and other) variants of P Systems, and [7] for a complete list of references to the bibliography of P Systems.

In this paper we present another extension of P Systems, called P Systems with Membrane Channels (PMC Systems), in which the passage of objects through membranes is not always allowed. In particular, membranes are associated with channels that enable the passing of objects through the membrane itself. Objects can pass through a membrane only if there are channels on the membrane that enable such a passage.

The definition of this extension of P Systems has a biological inspiration. In fact, the passing of proteins and other molecules through cell membranes is usually made possible by other proteins, actually called membrane channels or membrane transport proteins, which are placed on membrane surfaces. Such proteins either create a small hole through which molecules can freely pass (in this case they are called diffusion channels) or behave as a pump by binding to molecules on one side of the membrane, pushing them on the other side and releasing them (in this case they are called transport channels).

We show, with a simple example of a system computing 2^n, that PMC Systems may be more succinct than the P Systems computing the same functions. Moreover, we show that PMC Systems are universal even if only the simplest form of evolution rules is considered, namely non–cooperative rules. Finally, we give two application examples to show that membrane channels can ease the description of biological systems when PMC Systems are used as a modeling formalism, and we compare PMC Systems with other variants of P Systems.

2. P Systems with Membrane Channels

In this section we recall the definition of standard P Systems, and then we define their extension with membrane channels. We will denote multisets over a finite alphabet as strings of alphabet symbols. More precisely, let V^* be the set of all strings over an alphabet V, including the empty one, denoted by λ. For $a \in V$ and x in V^* we denote by $|x|_a$ the number of occurrences of a in x. If $V = \{a_1, \ldots, a_n\}$ (the ordering is important here), then the Parikh mapping of x is defined by $\Psi_V(x) = (|x|_{a_1}, \ldots, |x|_{a_n})$. The definition is extended in the natural way to languages. A string x represents the multiset over V with the multiplicities of objects $a_1, \ldots, a_n$ as given by $\Psi_V(x)$.

2.1. P Systems

A P System consists of a *hierarchy of membranes* that do not intersect, with a distinguishable membrane, called the *skin membrane*, surrounding them all. As usual, we assume membranes to be labeled by natural numbers. Given a set of objects V, a membrane m contains a multiset of *objects* in V^*, a set of *evolution rules*, and possibly other membranes, called *child* membranes (m is also called the *parent* of its child membranes). Objects represent molecules swimming in a chemical solution, and evolution rules represent chemical reactions that may occur inside the membrane containing them. For each evolution rule there is a multiset of objects representing the reactants, and a multiset of objects representing the products of the chemical reaction. A rule in a membrane m can be applied only to objects in m, meaning that the reactants should be precisely in m, and not in its child membranes. The rule must contain target indications, specifying the membranes where the new objects produced by applying the rule are sent. The new objects either remain in m, or can be sent out of m, or can be sent into one of its child membranes, precisely identified by its label. Formally, the products of a rule are denoted with a multiset of *messages* of the forms:

- $(v, here)$, meaning that the multiset of objects v produced by the rule remain in the same membrane m;
- (v, out), meaning that the multiset of objects v produced by the rule are sent out of m;
- (v, in_l), meaning that the multiset of objects v produced by the rule are sent into the child membrane l.

Let TAR be the set of message targets $\{here, out\} \cup \{in_i \mid i \in \mathbb{N}\}$. Given a set of objects O we denote with O_{tar} the corresponding set of messages $O \times TAR$. Hence, we denote with V_{tar} the set of all messages and we can define an evolution rule as a rule $u \to v$ such that $u \in V^*$ and $v \in V_{tar}^*$. The size of the left–hand side u of an evolution rule is called the *radius* of such a rule. If a P System contains rules of radius greater than one, then it is called a *cooperative* system. Otherwise, it is called *non–cooperative*.

Application of evolution rules is done with maximal parallelism, namely at each evolution step a multiset of instances of evolution rules is chosen non–deterministically such that no other rule can be applied to the system obtained by removing all the objects necessary to apply all the chosen rules.

A P System has a tree–structure in which the skin membrane is the root and the membranes containing no other membranes are the leaves. We assume membranes labels to be unique. A membrane structure can be represented as a balanced sequence of labeled brackets and, graphically, as a Venn diagram.

Definition 2.1. A *P System* is a tuple $\Pi = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$, where:

- V is a finite *alphabet* whose elements are called *objects*;
- $\mu \subset \mathbb{N} \times \mathbb{N}$ is a *membrane structure*, such that $(i, j) \in \mu$ denotes that the membrane labeled by j is contained in the membrane labeled by i;
- w_i with $1 \leq i \leq n$ are strings from V^* representing multisets over V associated with membranes $1, 2, \ldots, n$ of μ;
- R_i with $1 \leq i \leq n$ are finite sets of *evolution rules* associated with membranes $1, 2, \ldots, n$ of μ.

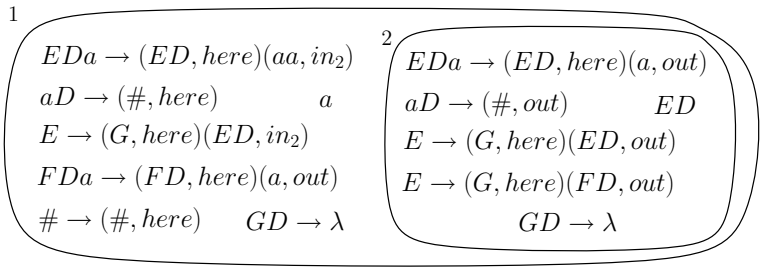

Figure 1. Example of P System that computes $\{a^{2^n} \mid n \in \mathbb{N}\}$.

A sequence of transitions between configurations of a given P System Π is called a *computation*. A computation is *successful* if and only if it reaches a configuration in which no rule is applicable. The result of a successful computation is the multiset of objects sent out of the skin membrane during the computation. Unsuccessful computations (computations which never halt) yield no result. Given a P System Π whose set of object is V, the result $x \in V^*$ of a computation of Π can be represented as the vector of natural numbers $\Psi_V(x)$. The set of all vectors of natural numbers computed by Π is denoted $Ps(\Pi)$.

In Figure 1 we show an example of P System Π_1 computing $\{a^{2^n} \mid n \in \mathbb{N}\}$, namely such that $Ps(\Pi_1) = \{2^n \mid n \in \mathbb{N}\}$. Initially, only in membrane 2 there are rules which are applicable and send either objects F and D or objects E and D into membrane 1. In the former case the object a in membrane 1 is sent out and the computation halts. In the latter case object a in membrane 1 is consumed and two occurrences of a are sent into membrane 2. Subsequently, E is consumed and sent into membrane 2 together with D. Note that the rule which sends ED into membrane 2 cannot be applied while there are still objects a in membrane 1, otherwise by the maximal parallelism also the rule producing $\#$ would be applied giving rise to an infinite (unsuccessful) computation. Objects a sent into membrane 2 are then sent back into membrane 1. The process of doubling and sending into membrane 2 we have explained, could be repeated an arbitrary number of times. Note that all the rules consuming a act on a single occurrence of a at a time and hence the time complexity of the computation is proportional to 2^{n+2}.

2.2. Extension with Membrane Channels

The difference between standard P Systems and PMC Systems is that in the latter ones communication of objects through membranes is not always allowed. In addition to standard P Systems, a membrane in a PMC System has associated a multiset of *transport channels* and a set of *diffusion channels*. A transport channel, denoted as an object $a \in V$, enables the passing through the membrane of a single occurrence of object a for each computation step. A diffusion channel, denoted $\overline{a}$ with $a \in V$, enables the passing through the membrane of an arbitrary number of objects a for each computation step. Note that transport channels associated with a membrane are a multiset rather than a set because multiple occurrences of the same channel may enable the passing of more than one occurrences of the same object at the same time. Channels are said to be *oriented* when they enable the passing of objects in one only direction, either from inside a membrane to outside, or vice-versa. An oriented channel is prefixed by $\uparrow$ when it allows objects to exit a membrane, and it is prefixed by $\downarrow$ when it allows objects to enter a membrane. The set

of all channels is given by $V_{ch} = \{a, \uparrow a, \downarrow a, \overline{a}, \uparrow\overline{a}, \downarrow\overline{a} \mid a \in V\}$.

Objects can pass through a membrane only if there are some channels enabling the passage. This means that the presence of channels influences the applicability of the evolution rules sending objects either outside or into some inner membrane. An evolution rule sending an object through a membrane can be applied only if the membrane has a channel enabling this operation. If two evolution rules send the same object through the same membrane they can be applied together only if the membrane has either a diffusion channel or at least two transport channels enabling these operations. If the membrane has a single transport channel, one of the two rules is chosen non–deterministically.

An evolution rule can modify the channels on a membrane by adding or deleting them. Formally, the products of a rule are enriched with a multiset of *channel modifications*. Channel modifications are of the forms:

- $\langle +c, -c' \rangle_{here}$, meaning that the multisets of channels c and c' are added to and removed from the membrane containing the evolution rule itself, respectively;
- $\langle +c, -c' \rangle_{on_l}$, meaning that the multisets of channels c and c' are added to and removed from the child membrane l, respectively.

With abuse of notation, we denote c and c' as multisets in V_{ch}^* even if we do not allow them to contain multiple occurrences of a diffusion channel. Note that we use on_l as subscript of channel modifications rather than in_l (that is used in messages of evolution rules) to emphasize the fact that channels model proteins that are placed on membrane surfaces rather than inside membranes.

Channel modifications do not influence the applicability of an evolution rule. In particular, evolution rules which remove a channel that is not present can be applied anyway. In this case the channel modification will have no effect.

Definition 2.2. A *PMC System* is a tuple $(V, \mu, w_1, \ldots, w_n, c_1, \ldots, c_n, R_1, \ldots, R_n)$ where:

- V is an *alphabet* whose elements are called *objects*;
- $\mu \subset \mathbb{N} \times \mathbb{N}$ is a *membrane structure*, such that $(i, j) \in \mu$ denotes that the membrane labeled by j is contained in the membrane labeled by i;
- w_i with $1 \leq i \leq n$ are strings from V^* representing multisets over V associated with membranes $1, 2, \ldots, n$ of μ;
- c_i with $1 \leq i \leq n$ are strings from V_{ch}^* representing multisets over V_{ch} associated with membranes $1, 2, \ldots, n$ of μ. In these strings diffusion channels cannot appear more than once;
- R_i with $1 \leq i \leq n$ are finite sets of *evolution rules* associated with membranes $1, 2, \ldots, n$ of μ.

The notions of (successful) computation and of result of computations of PMC Systems are the same as for standard P Systems.

In Figure 2 we show an example of PMC System Π_2 computing, as the P System Π_1 in Section 2.1, $\{a^{2^n} \mid n \in \mathbb{N}\}$, namely such that $Ps(\Pi_2) = Ps(\Pi_1) = \{2^n \mid n \in \mathbb{N}\}$. The multiset of channels initially associated with each membrane appears in the figure together with the membrane label. Initially, the only two applicable rules are those which consume E in membrane 1: the first opens a diffusion

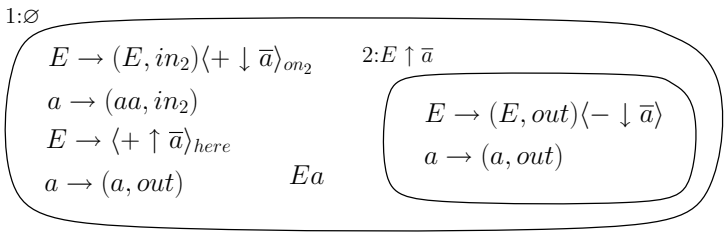

Figure 2. Example of PMC System that computes $\{a^{2^n} \mid n \in \mathbb{N}\}$.

channel allowing a to be sent out and the second opens a diffusion channel allowing a to be sent into membrane 2. Subsequently, in the former case a is sent out, and the computation halts. In the latter case a is doubled and sent into membrane 2. Membrane 2 closes the previously opened channel and sends all objects a back into membrane 1. The process could be repeated an arbitrary number of times. With respect to Π_1 we use here only non-cooperative rules and the time complexity is proportional to n.

3. Universality of PMC Systems

In this section we prove a universality result for PMC Systems by showing that any matrix grammar with appearance checking can be simulated by a PMC System. As a consequence, before giving the result and its proof, we recall from [5] the definition of such variant of matrix grammars and some related notions.

3.1. Matrix grammars with appearance checking

A (context-free) matrix grammar with appearance checking is a tuple $G = (N, T, S, M, F)$, where N and T are disjoint alphabets of non–terminals and terminals, respectively, $S \in N$ is the axiom, M is a finite set of matrices, namely sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$ of context–free rules over $N \cup T$ with $n \geq 1$, and F is a set of occurrences of rules in the matrices of M. For a string w, a matrix $m : (r_1, \ldots r_n)$ can be executed by applying its rules to w sequentially in the order in which they appear in m. Rules of a matrix occurring in F can be skipped during the execution of the matrix if they cannot be applied, namely if the symbol in their left–hand side is not present in the string.

Formally, given $w, z \in (N \cup T)^*$, 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 $w_i \in (N \cup T)^*$ with $1 \leq i \leq n+1$ such that $w = w_1$, $z = w_{n+1}$ and, for all $1 \leq i \leq n$, either (1) $w_i = w'_i A_i w''_i$ and $w_{i+1} = w'_i x_i w''_i$, 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. We remark that F consists of *occurrences* of rules in M, that is, if the same rule appears several times in the matrices, it is possible that only some of these occurrences are contained in F.

The language generated by a matrix grammar with appearance checking G is defined as $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$, where $\Longrightarrow^* w$ is the reflexive and transitive closure of $\Longrightarrow$. The family of languages of this form is denoted by MAT^{λ}_{ac}, when rules having the empty string λ as right hand side (λ–rules) are allowed, and by MAT_{ac} when such rules are not allowed. Moreover, the family of languages generated by matrix grammars without appearance checking (i.e. with $F = \varnothing$) is denoted by MAT^{λ}, when λ–rules are allowed, and by MAT, when such rules are not allowed. It is known (see [5]

for details) that (i) $MAT \subset MAT_{ac} \subset CS$; (ii) $MAT^{\lambda} \subset MAT^{\lambda}_{ac} = RE$, where CS and RE are the families of languages generated by context–sensitive and arbitrary grammars, respectively.

Let $ac(G)$ be the cardinality of F in G and let $|x|$ denote the length of the string x. A matrix grammar with appearance checking $G = (N, T, S, M, F)$ is said to be in the *strong binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these sets mutually disjoint, and the matrices in M are in one of the following forms:

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

Moreover, there is only one matrix of type 1, F consists exactly of all rules $A \to \#$ appearing in matrices of type 3 and $ac(G) \leq 2$. We remark that $\#$ is a trap symbol, namely once introduced it cannot be removed, and a matrix of type 4 is used only once, in the last step of a derivation.

For each matrix grammar (with or without appearance checking) there exists an equivalent matrix grammar in the strong binary normal form. Consequently, for each language $L \in RE$ there exists a matrix grammar with appearance checking G satisfying the strong binary normal form and such that $L(G) = L$.

Conventions A matrix grammar with appearance checking in the strong binary normal form is always given as $G = (N, T, S, M, F)$, with $N = N_1 \cup N_2 \cup \{S, \#\}$ and with $n + 1$ matrices in M, injectively labeled with $m_0, m_1, \ldots, m_n$. The matrix $m_0 : (S \to X_{init} A_{init})$ is the initial one, with X_{init} a given symbol from N_1 and A_{init} a given symbol from N_2; the next k matrices are without appearance checking rules, $m_i : (X \to \alpha, A \to x)$, with $1 \leq i \leq k$, where $X \in N_1, \alpha \in N_1 \cup \{\lambda\}, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$ (if $\alpha = \lambda$, then $x \in T^*$); the last $n - k$ matrices have rules to be applied in the appearance checking mode, $m_i : (X \to Y, A \to \#)$, with $k + 1 \leq i \leq n, X, Y \in N_1$, and $A \in N_2$.

Since the grammar is in the strong binary normal form, we have (at most) two symbols $B^{(1)}$ and $B^{(2)}$ in N_2 such that the rules $B^{(j)} \to \#$ appear in matrices m_i with $k + 1 \leq i \leq n$. For $j \in \{1, 2\}$, we denote with ℓ_j the set $\{i \mid$ the matix m_i contains the rule $B^{(j)} \to \#\}$. For uniformity, we also denote $\ell_0 = \{1, 2, \ldots, k\}$ and $\ell = \ell_0 \cup \ell_1 \cup \ell_2 = \{1, 2, \ldots, n\}$ (note that $0 \notin \ell$). Clearly, the sets ℓ_0, ℓ_1 and ℓ_2 are disjoint.

We remark that in matrix grammars in strong binary normal forms we can assume that all symbols $X \in N_1$ appear as the left-hand side of a rule from a matrix: otherwise, the derivation is blocked after introducing such a symbol, hence we can remove these symbols and the matrices involving them.

3.2. Universality

We prove that PMC Systems are universal by showing that the family, denoted $PsMC_4(ncoo)$, of sets $Ps(\Pi_{mc})$ of results computed by PMC Systems with at least four membranes and with non–cooperative rules is equivalent to the family, denoted $PsRE$, of the images of all the languages in RE obtained through the Parikh mapping (this is the family of recursively enumerable sets of vectors of natural numbers). As P Systems with non-cooperative rules are not universal, our result implies that universality is due to the presence of membrane channels.

Theorem 3.1. $PsMC_4(ncoo) = PsRE$.

Proof:
It is enough to show that for a G in strong binary normal form there is a PMC System Π_G such that $Ps(\Pi_G) = \Psi_T(L(G))$. We build Π_G as a system with a root membrane, labeled 3, with three child membranes, labeled 0, 1 and 2. The alphabet contains an object for each element in $N \cup T$, an object λ_i for every $i \in \ell$, and a *token* object such that only the membrane with the token can run. In the initial configuration, membranes 0 and 3 are associated with an empty multiset of channels, while membranes 1 and 2 are associated with the multiset $(\bigcup_{A \in N_2} \uparrow \overline{A})(\bigcup_{Y \in N_1} \uparrow Y) \uparrow E$. Membrane 3 has initially the token and the objects corresponding to X_{init} and A_{init}. It has a cyclic behavior. It selects an index $i \in \{0, 1, 2\}$ and orders to membrane i to simulate a matrix with label in ℓ_i. To this purpose, 3 sends to i the token and the objects corresponding to the two nonterminals in the left side of the two rules of the selected matrix. Membrane i exploits these objects to simulate the two rules and, then, returns to 3 the token and the objects corresponding to the products of the rules, if any. Those corresponding to terminals are sent out by 3, which can restart the cycle once more. When 3 has no more objects corresponding to nonterminals, the computation ends, and the multiset of the objects that have been sent out correspond to a multiset in $\Psi_T(L(G))$. Simulating a sequence of matrices that either cause an infinite loop in G or originate nonterminals that cannot be removed, gives rise to an infinite loop by Π_G. This infinite loop starts when the *trap* object $\#$ is produced, and will be called *trap loop*. Consequently, $Ps(\Pi_G) = \Psi_T(L(G))$. The rules in membrane 3 are the following:

1. $\{X \to (\alpha_i, here)\langle + (\bigcup_{A \in N_2} \downarrow \overline{A}) \downarrow \alpha_i \downarrow A_i \downarrow E \rangle_{on_0} | m_i = (X \to \alpha, A \to x), i \in \ell_0\}$. Given any matrix $m_i = (X \to \alpha, A \to x)$ with $i \in \ell_0$, if membrane 3 has the object X needed to fire the first rule, then it can decide to ask membrane 0 to simulate m_i. To this purpose, it opens the channels needed to send to 0 all nonterminals in N_2, one occurrence of both α and A marked with i, and the token E. The task of membrane 0 will be to exploit A_i to produce x. The reason for which all other nonterminals in N_2 are sent to 0 will be clarified at point 11. If, incorrectly, no A marked with i will be sent to 0, which corresponds to fire only the first rule of the matrix, membrane 0 will detect this error and enter a trap loop.

2. $\{X \to (Y_i, here)\langle + (\bigcup_{A \in N_2} \downarrow \overline{A}) \downarrow Y_i \downarrow E \rangle_{on_1} | m_i = (X \to Y, B^{(1)} \to \#), i \in \ell_1\}$. This case is similar to case 1. Marking $B^{(1)}$ with i is not needed here, since membrane 1 is not required to exploit $B^{(1)}$ to produce anything. The task of 1 shall be to check that $B^{(1)}$ is not available, so that applying the first rule of the matrix $(X \to Y, B^{(1)} \to \#)$ is legal.

3. $\{X \to (Y_i, here)\langle + (\bigcup_{A \in N_2} \downarrow \overline{A}) \downarrow Y_i \downarrow E \rangle_{on_2} | m_i = (X \to Y, B^{(2)} \to \#), i \in \ell_2\}$. This case is like case 2.

4. $E \to (E'E'', here)\langle - \uparrow ck \rangle_{here}$. When membrane 3 has the token E, it replaces it with objects E' and E'', and closes its output channel ck. Notice that this rule is always performed together with exactly one of the rules considered in the cases 1, 2, 3.

5. $\{E' \to (E, in_i) \,|\, 0 \leq i \leq 2\}$. The object E' generated with rule 4 is exploited to give the token to one of the child membranes. Notice that only one child has the input channel E open and can receive the token, since only one rule among those considered in cases 1, 2 and 3 has fired.

6. $E'' \to \langle + \uparrow ck \rangle_{here}$. The object E'' generated with rule 4 is used to open the output channel ck.

7. $\{\alpha_i \to (\alpha_i, in_0), A \to (A_i, in_0) | m_i = (X \to \alpha, A \to x), i \in \ell_0\}$. The channels opened by firing one of the rules considered in 1 are exploited to send both one A_i and one α_i to membrane 0.

8. $\{Y_i \to (Y_i, in_1) | m_i = (X \to Y, B^{(1)} \to \#), i \in \ell_1\}$. Analogous to case 7.

9. $\{Y_i \to (Y_i, in_2) | m_i = (X \to Y, B^{(2)} \to \#), i \in \ell_2\}$. Analogous to case 7.

10. $\{A \to (A, in_i) | A \in N_2, 0 \leq i \leq 2\}$. The channels opened with the rule that has fired among those considered in 1, 2, 3, are exploited to send all unmarked nonterminals to membrane 0, or 1, or 2.

11. $A \to (\#, here)(ck, out)$. This rule can be applied only when the output channel ck has been opened with the rule considered in 6, and only if some $A \in N_2$ is available and has not been sent to any child. Since all objects in N_2 have already been sent by the rules considered in 10 if the input channels of some child membrane were opened, we infer that this rule is fired only if these channels are all closed. This can happen only when no nonterminal in N_1 is available and no rule among those considered in 1, 2, 3 has fired. Since having A together with no nonterminal in N_1 implies that G cannot generate any string of terminals, this rule generates the trap symbol $\#$.

12. $\{a \to (a, out) | a \in T\}$. All terminals are sent out.

13. $\# \to (\#, here)$. This is the trap rule.

14. $d \to \lambda$. Object d can be received from membrane 0. Such a communication is exploited by membrane 0 but object d is useless here and is removed.

The rules of membrane 0 are the following:

1. $E \to (E', here)\langle -(\bigcup_{A \in N_2} \downarrow \overline{A})(\bigcup_{Y \in N_1, i \in \ell_0} \downarrow Y_i)(\bigcup_{A \in N_2, i \in \ell_0} \downarrow A_i)(\bigcup_{i \in \ell_0} \downarrow \lambda_i) \downarrow E \rangle_{here}$. Upon receiving the token, membrane 0 closes its input channels and generates the object E'.

2. $\{A_i \to (x, here)\langle + \uparrow d \rangle_{here} | m_i = (X \to \alpha, A \to x), i \in \ell_0\}$. Object A_i is exploited to generate x, as required by matrix m_i, and to open the output channel d. This rule is fired together with that in 1.

3. $E' \to (E'', here)(d, out)\langle +(\bigcup_{A \in N_2} \uparrow \overline{A})(\bigcup_{Y \in N_1} \uparrow Y)(\bigcup_{a \in T} \uparrow a) \uparrow E \rangle_{here} \langle - \uparrow d \rangle_{here}$. Object E' generated by the rule in 1 is exploited to open all output channels needed to send all terminals and nonterminals to membrane 0, and to close the output channel d. This rule can be applied only if the output channel has been opened by the rule in 2.

4. $E' \to (\#, here)$. This rule forces the membrane to enter the trap loop. To avoid the loop, E' must be consumed by the rule considered in 3. This can happen only if the output channel d has been opened in rule 2. In turn, this is possible only if one A_i has been received from membrane 3. Summarizing, this avoids that membrane 3 applies the first rule of some matrix $(X \to \alpha, A \to x)$ without asking to membrane 0 to apply the second.

5. $\{Y_i \to (Y, out) | Y \in N_1\} \cup \{A \to (A, out) | A \in N_2\} \cup \{a \to (a, out) | a \in T\}$. Channels opened with rule 3 are exploited to send all terminals and nonterminals to membrane 0.
6. $E'' \to (E, out)\langle -(\bigcup_{A \in N_2} \uparrow \overline{A})(\bigcup_{Y \in N_1} \uparrow Y)(\bigcup_{a \in T} \uparrow a) \uparrow E\rangle_{here}$. The object E'' generated with the rule in 3 is exploited to send the token to membrane 3 and close all the output channels.
7. $\# \to (\#, here)$.
8. $\lambda_i \to \lambda$. Membrane 0 does not need λ_i.

The rules for membrane 1 are the following:

1. $E \to (E, out)\langle -(\bigcup_{A \in N_2} \downarrow \overline{A})(\bigcup_{Y \in N_1, i \in \ell_1} \downarrow Y_i) \downarrow E\rangle_{here}$. Upon receiving the token, membrane 1 closes its input channels and sends the token back.
2. $B^{(1)} \to (\#, here)$. At the same step, if $B^{(1)}$ is present, the trap loop starts.
3. $\{Y_i \to (Y, out) | Y \in N_1\} \cup \{A \to (A, out) | A \in N_2\}$. All nonterminals are immediately returned to membrane 3.
4. $\# \to (\#, here)$.

The rules for membrane 2 are the same, with ℓ_2 and $B^{(2)}$ replacing ℓ_1 and $B^{(1)}$, respectively □

4. Applications

In this section we give two applications of P systems with transport and diffusion membrane channels to the specification of biological systems. The first one is the specification of the lactose operon in *Escherichia coli*, and the second one is the specification of the initial phases of the EGFR signalling cascade.

The specification of lactose operon uses only transport channels, while in the EGFR specification also diffusion channels are necessary.

4.1. The lactose operon in E. coli

We give a PMC system modeling the regulation process of the lactose operon in *Escherichia coli*. The lactose operon is a sequence of six genes that are responsible for producing three enzymes for lactose degradation, namely the *lactose permease*, which is incorporated in the membrane of the bacterium and actively transports the sugar into the cell, the *beta galactosidase*, which splits lactose into glucose and galactose, and the *transacetylase*, whose role is marginal. The first three genes of the operon (i,p,o) regulate the production of the enzymes, and the last three (z,y,a), called *structural genes*, are transcribed (when allowed) into the mRNA for beta galactosidase, lactose permease and transacetylase, respectively.

The regulation process is as follows (see Figure 3): gene i encodes the *lac repressor*, which, in the absence of lactose, binds to gene o (the *operator*). Transcription of structural genes into mRNA is performed by the RNA polymerase enzyme, which usually binds to gene p (the *promoter*) and scans the operon from left to right by transcribing the three structural genes z, y and a into a single mRNA fragment. When the lac repressor is bound to gene o, it becomes an obstacle for the RNA polymerase,

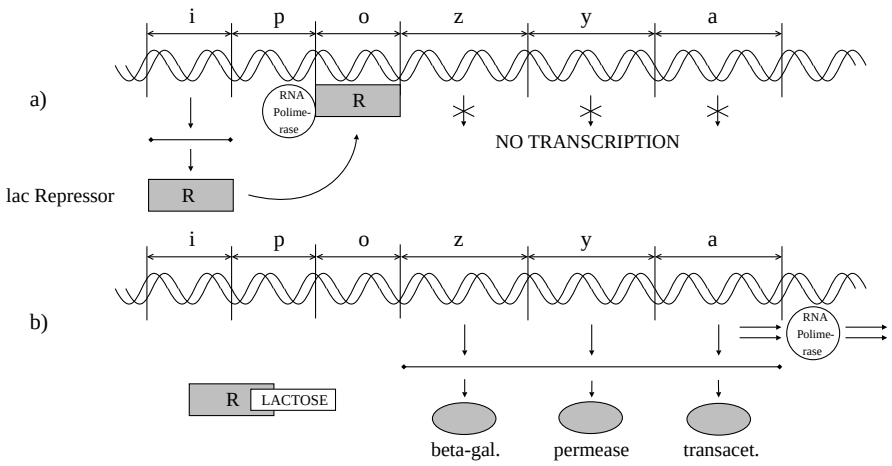

Figure 3. The regulation process in the Lac Operon.

and transcription of the structural genes is not performed. On the other hand, when lactose is present inside the bacterium, it binds to the repressor and this cannot stop anymore the activity of the RNA polymerase. In this case the transcription is performed and the three enzymes for lactose degradation are synthesized.

We construct a PMC System modeling the regulation process by ignoring, for the sake of simplicity, the production of the beta galactosidase and of the transacetylase enzymes, and the production of the RNA as an intermediate step in the production of the enzymes. The model we construct is the PMC System

$$\Pi_{lac} = (V, [_1 [_2]_2]_1, lact^n, lacI\, lacP\, lacO\, polym, \varnothing, \varnothing, R_1, R_2)$$

where $V = \{lact, lacI, lacP, lacO, polym, PlacP, perm, repr, RO, rlact, permdeg\}$ and the sets of rules R_1 and R_2 are as follows:

$$R_1 = \{\, lact \to (lact, here)\,,\ lact \to (lact, in_2)\,\}$$

$$\begin{gathered} R_2 = \{\, lacI \to (lacI\, repr, here)\,,\ repr\, lacO \to (RO, here)\,,\ RO \to (repr\, lacO, here)\,, \\ polym\, lacP \to (PlacP, here)\,,\ PlacP\, lacO \to (lacP\, lacO\, polym\, perm, here)\,, \\ repr\, lact \to (rlact, here)\,,\ rlact \to (repr\, lact, here)\,, \\ perm \to (permdeg, here)\, \langle + \downarrow lact \rangle_{here}\,, \\ repr \to \lambda\,,\ perm \to \lambda\,,\ permdeg \to \langle - \downarrow lact \rangle_{here}\,\} \cup \{a \to (a, here) \mid a \in V\} \end{gathered}$$

The membrane structure of the PMC System Π_{lac} consists of two membranes contained one inside the other. Membrane 1 represents the environment which may contain lactose. By assuming $lact^n$ in the initial configuration we mean that there are a number of molecules $n \geq 0$. Membrane 2 represents a bacterium which contains genes i, p and o (denoted $lacI, lacP$ and $lacO$, respectively) and the RNA polymerase enzyme (denoted $polym$). Initially, there are channels neither in membrane 1 nor in membrane 2. R_1 and R_2 are the sets of rules in membrane 1 and membrane 2, respectively. By the application of the rules of R_1 a number of molecules of lactose may enter membrane 2 if suitable channels

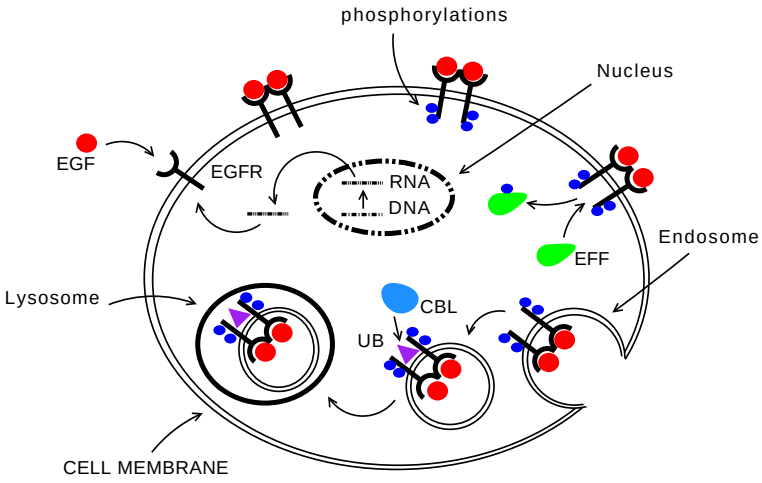

Figure 4. The EGF signalling pathway.

exist on membrane 2. The first five rules of R_2 describe the production of the repressor (denoted $repr$), the reversible binding of the repressor with gene o (the complex is denoted RO), and the production of permease (denoted $perm$). The sixth and the seventh rules describe the reversible binding of repressor and lactose (the complex is denoted $rlact$). The rule $perm \to permdeg\,\langle + \downarrow lact\rangle_{here}$ describes the incorporation of the permease enzyme in the membrane of the bacterium. This is modeled by the opening of a transport channel on membrane 2 that allows lactose to enter. The rule produces an object $permdeg$ which is used in the rule that models the degradation of permease enzymes on the membrane by removing occurrences of transport channels. The rules of the set $\{a \to (a, here) \mid a \in V\}$ have been introduced to have a more natural form of parallelism which is not the maximal one assumed in P Systems.

4.2. The EGF signalling pathway

In Biology, signal transduction refers to any process by which a cell converts one kind of signal or stimulus into another. Signals are typically proteins that may be present in the environment of the cell. In order to be able to receive the signal, namely to recognize that the corresponding protein is available in the environment, a cell exposes some receptors on its external membrane. A receptor is a transmembrane protein that can bind to a signal protein on its extracellular end. When such a binding is established, the intracellular end of the receptor undergoes a conformational change that enables interaction with other proteins inside the cell. This typically causes an ordered sequence of biochemical reactions inside the cell, usually called signalling pathway, that are carried out by enzymes and may produce different effects on the cell behaviour.

A complex signal transduction cascade, that modulates cell proliferation, survival, adhesion, migration and differentiation, is based on a family of receptors called epidermal growth factor receptors (EGFRs). While EGFR signalling is essential for many normal morphogenic processes, the aberrant activity of these receptors has been shown to play a fundamental role in proliferation of tumor cells. Epi-

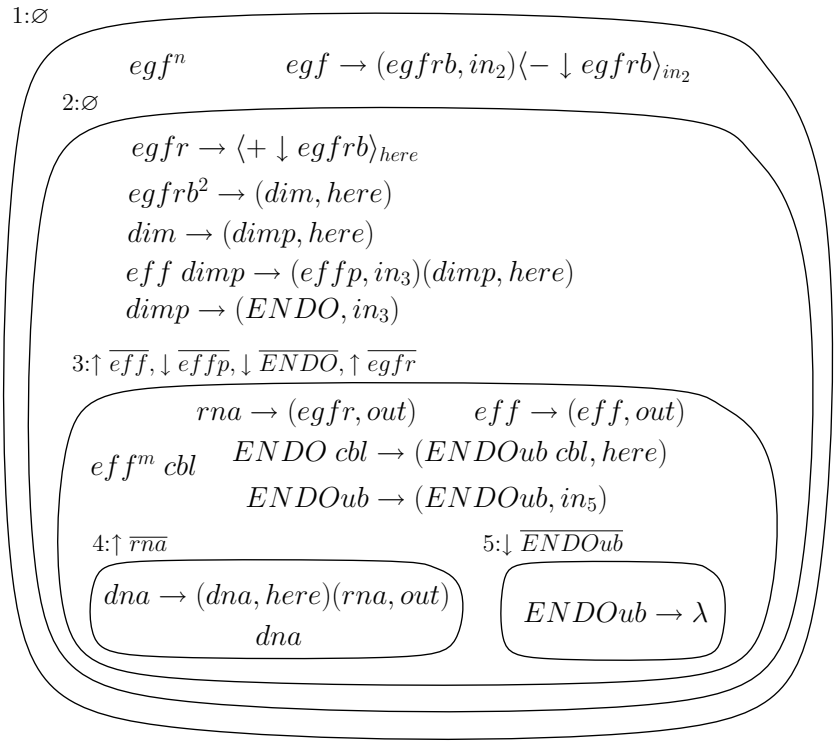

Figure 5. A P system with membrane channels model of the EGF signalling pathway.

dermal growth factor receptors are produced by specific genes in the DNA (through the RNA) and they are located on the cell surface. Receptors are activated by the binding with a specific ligand (epidermal growth factor, EGF) to form a EGFR (ligand-receptor) complex. Upon activation, EGFR undergoes a transition from a monomeric form to an active dimeric one. EGFR dimerization stimulates its intracellular phosphorylation which activates signalling proteins. These activated signalling proteins (effector proteins) initiate several signal transduction cascades, leading to DNA synthesis and cell proliferation. After the activation of effector proteins, ligand-receptor dimers are internalized in endosomes. An ubiquitin ligase, known as Cbl, binds an ubiquitin protein to the dimer (ubiquitination). The ubiquitin protein targets the dimers for lysosomal degradation (see Figure 4).

The PMC system modeling the EGF is given in Figure 5.

Membrane 1 models the environment external to the cell, membrane 2 represents the cell surface, membrane 3 models the inner of the cell, membrane 4 is the nucleus, and membrane 5 represents a lysosome.

In the external environment egf corresponds to the epidermal growth factor EGF which can bind the receptor on the surface of the cell. The receptor is modeled by $egfr$ in membrane 2, which can open a transport channel for $egfrb$. The complex of egf with the receptor is simulated by the passing of the element $egfrb$, generated by egf, through the channel generated by $egfr$. After the binding the transport channel is closed so that the number of generated complexes cannot be greater than the number of receptors.

Two $egfrb$s on the surface of the cell can dimerize (dim), and the dimer can be phosphorilated ($dimp$). The phosphorilated dimer can activate the effector eff, coming from inside the cell (membrane 3), producing $effp$ which is sent inside the cell itself. The phosphorilated dimer can also be enclosed in an endosome (ENDO) which goes in the cell cytoplasm.

The three elements eff, $effp$ and $ENDO$ can pass freely from inside the cell to the cell surface (and vice versa) through the corresponding diffusion channels.

The nucleus of the cell (membrane 4) is responsible for the production of $egfr$ through the DNA and RNA (dna and rna). The rna reaches the cell cytoplasm, through the corresponding diffusion channel on the nucleus membrane, and there it produces $egfr$ which is sent on the cell surface.

Finally a dimer enclosed in an endosome is targeted by the ubiquitin cbl ($ENDOub$) and it is sent inside a lysosome (membrane 5) where it is destroyed.

Note that, for the sake of compactness, we have omitted to write, in each membrane, the rules of the form $a \to (a, here)$, for all $a \in V$ belonging the membrane itself. As said before, such rules allow to assume a more natural form of parallelism than the maximal one.

5. Related work and conclusions

Membrane channels have been already investigated in [3]. In this paper a variant of purely communicating P Systems [6] is considered where multisets of activators can open channels for certain objects to pass through membranes in one direction. Prohibitors can control the permeability of channels by forbidding objects to pass. It is shown that singleton activators and prohibitors are enough to attain universality. The translation of this formalism into PMC Systems would give an alternative proof of universality of PMC Systems.

In [2] a variant of P Systems is defined in which transport of objects across the regions of the system is possible by means of rules associated with membranes and involving proteins attached to them. Such proteins can be attached/de-attached to/from membranes by means of rules. This makes it possible to express more complex conditions on transport of objects through membranes. However, as the assumption of maximal parallelism is released, universality is not attained and decidability of reachability of configurations is proved.

In this paper we have proposed a variant of P Systems with transport and diffusion membrane channels, and we have proved a universality result. We have claimed by means of an example that the formalism may allow more succinct descriptions. We have also given two examples of application to biological systems, in which we use both transport and diffusion channels. In the examples the role of transport channels which can control the number of objects passing through membranes, and the role of diffusion channels which allow objects to cross freely the membrane boundaries, are clear.

Acknowledgments

Work partially supported by MiUR PRIN 2006 project “Biologically Inspired Systems and Calculi and their Applications (BISCA)”.

References

[1] Bottoni, P., Martin–Vide, C., Păun, G., Rozemberg., G.: Membrane systems with promoters/inhibitors. *Acta Informatica* **38**, 2002, 695–720.

[2] Cavaliere, M., Seawards, S.: Membrane systems with peripheral proteins: transport and evolution. *Proc. Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'06)*, ENTCS **171**, 2007, 37–53.

[3] Freund, R., Oswald, M.: P systems with activated/prohibited membrane channels, *Proc. Workshop on Membrane Computing (WMC 2002)*, LNCS 2597, 2003, 261–269.

[4] Păun, G.: Computing with membranes, *Journal of Computer and System Sciences* **61**, 2000, 108–143.

[5] Păun, G.: *Membrane Computing. An Introduction*, Springer, 2002.

[6] Păun, A., Păun, G.: The power of communication: P systems with symport/antiport, *New Generation Computing* **20**, 2002, 295–306.

[7] P Systems, web page. `http://ppage.psystems.eu/`.