P Systems with Endosomes

Roberto Barbuti, Giulio Caravagna, Andrea Maggiolo-Schettini, Paolo Milazzo

Università di Pisa
Dipartimento di Informatica
Largo Pontecorvo 3, 56127 Pisa, Italy
E-mail: {barbuti, caravagn, maggiolo, milazzo}@di.unipi.it

> **Abstract:** P Systems are computing devices inspired by the structure and the func-
> tioning 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 Endosomes (PE Systems), in which endosomes can be explicitly
> modeled. We show that PE Systems are universal even if only the simplest form of
> evolution rules is considered, and we give one application example.
> **Keywords:** P systems, PE Systems, Endosomes

1 Introduction

P Systems were introduced by Păun in [10] 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
membrane, where each membrane is identified by its labels, others are sent *out* of the membrane, others
are sent *into* the inner membranes. 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 surrounding
membrane 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.
See [11] for the definition of these (and other) variants of P Systems and [14] 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 Endosomes (PE
Systems), with the following features:

- objects can be contained inside the regions delimited by the membranes and on the surfaces of the
  membranes (as in P Systems with peripheral proteins [6, 13] and as in membrane systems with
  surface objects [1, 2]);

- rules are contained on the surfaces of the membranes (they can rewrite objects outside/on/into the membranes);
- endosomes can be explicitly created in order to model a biologically inspired transportation mechanism.

The definition of this extension of P Systems has a biological inspiration. In fact, the *endocytosis* of macromolecules is the process by which cells absorb material (molecules such as proteins) from outside the cell by engulfing it with their cell membrane. It is used by all cells because most substances important to them are large polar molecules that cannot pass through the hydrophobic plasma membrane or cell membrane. There exist three kinds of endocytosis: *phagocytosis*, *pinocytosis*, and *receptor–mediated endocytosis*. In particular, phagocytosis (literally, cell–eating) is the process by which cells ingest large objects, such as cells which have undergone apoptosis, bacteria, or viruses. The membrane folds around the object, and the object is sealed off into a large vacuole known as a phagosome. Pinocytosis (literally, cell–drinking) is concerned with the uptake of solutes and single molecules such as proteins, and, finally, receptor–mediated endocytosis is a more specific active event where the cytoplasm membrane folds inward to form coated pits. These inward budding vesicles bud to form cytoplasmic vesicles [11]. Figure 1 summarizes the kinds of endocytosis. From the point of view of the modeler, these three processes are made possible by vesicles (in fact, this transportation mechanism is known as *vesicle–mediated transportation*) which, in the most general case, engulf the macromolecules together with molecules from the surface of the membranes (i.e., receptors). This leads to the creation of *endosomes* containing the engulfed molecules. The endosomes transfer their content inside the cell by possibly interacting with other components. The endosomes could also be degraded by the interaction with the lysosomes. We define an extension of P Systems (PE Systems) which can explicitly model the creation of endosomes and their interaction inside the cells and, consequently, can easily model these three kinds of endocytosis.

This variant of P Systems, together with other modeling features such as the modeling of exocytosis (the biologically counterpart of endocytosis), and enriched with channel–mediated communication [3], would provide a powerful and complete modeling language for naturally describing transportation mechanism of molecules inside cells.
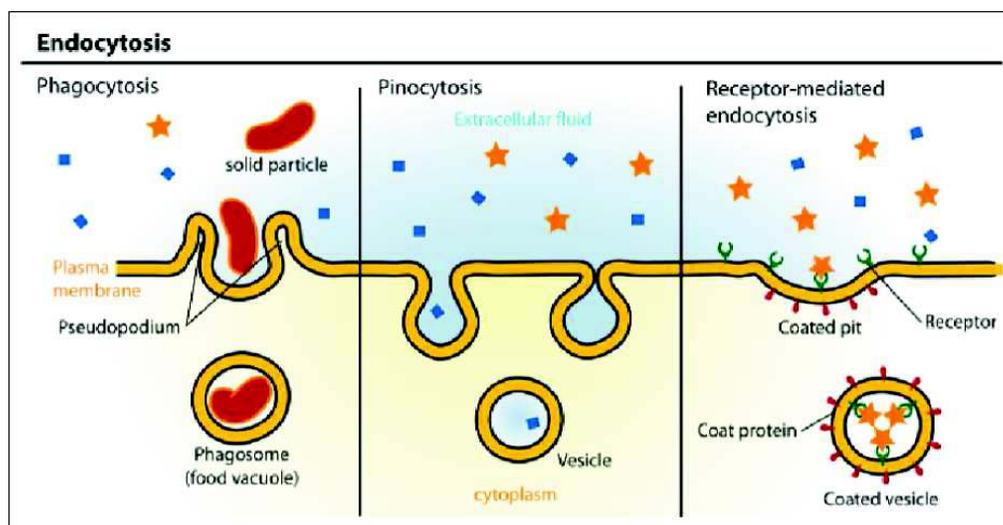

Figure 1: Three kind of endocytosis: *phagocytosis*, *pinocytosis* and *receptor–mediated endocytosis*. Picture taken from http://cellbiology.med.unsw.edu.au/units/science/lecture0806.htm

We show that PE Systems are universal even if only the simplest form of evolution rules is considered, namely non–cooperative rules. Finally, we give one application example to show that endosomes can ease the description of biological systems when PE Systems are used as a modeling formalism.

2   P Systems with Endosomes

In this section we formally define P Systems with endosomes (PE Systems). We assume the reader to be familiar with the standard definition of P Systems [11]. We start by assuming the same membrane structure μ of a P System. As regards objects, similarly to P Systems with peripheral proteins [6, 13], we assume that objects can be contained inside a membrane (as in classical P Systems) and on the surface of a membrane. In order to qualify in an evolution rule the position of an object with respect to a membrane, we use *in* to identify the object inside the membrane, *out* to identify the object outside the membrane and *here* to identify the object on the surface of the membrane. Let *TAR* be the set of message targets $\{in, out, here\}$; given a set of objects V we denote with V_{tar} the corresponding set of messages $O \times TAR$.

We can now introduce the evolution rules of PE Systems; rules are conceptually divided in evolution rules (in the same sense of P Systems) and rules for the creation of endosomes. We recall that, differently from P Systems, the rules of PE Systems are conceptually associated with the surfaces of the membranes of the system. Evolution rules are of the form $u \to v$ where $u \in V_{tar}^+$, $v \in V_{tar}^*$, $V_{tar}^+ = V_{tar}^* \backslash \{\varepsilon\}$ and ε is the empty string. The definition of cooperative and non–cooperative rules are the same as for P Systems.

Notice that this format for evolution rules, which are syntactically different from those of P Systems, may seem to be less expressive than the one of P Systems, in particular for rule moving objects into specific regions enclosed by membranes (communication rules). In order to show that this is not the case, let us assume an hypothetical membrane structure μ such that $(l, l') \in \mu$, namely a membrane structure in which l' is nested into l. In order to give a rule which moves an object inside membrane l' we cannot use the identifier $in_{l'}$ in a rule of the surface of the membrane l (as in P Systems) because we cannot use the identifier l' as subscript to *in*. However, the same behaviour can be obtained by replacing the rule $u \to (v, in_{l'})$ in the membrane l, as in usual P Systems, with the PE System rule $(u, out) \to (v, in)$ on the surface of the membrane l'. The behaviour modeled by this rule, which is in some sense an "attraction" by the nested membrane rather than the "sending" from the top membrane, leads to results analogous to those obtained by P Systems, namely to the transportation of the object inside the nested membranes.

The rules for creating endosomes are of the form $endo_E(u, v)$, where $u, v \in V^*$ and:

- E is a set of evolution rules for the endosome;
- u is the multiset of objects that must appear on the surface of the membrane containing the rule;
- v is the multiset of objects that must appear outside the membrane containing the rule.

Note that each endosome has got its own evolution rules in set E. These rules model the behaviour of the endosome. As regards the creation of an endosome, it is necessary that objects in u are present on the surface of the membranes (they can be seen as the receptors) and that objects in v are present outside of the membrane creating the endosome (they can be seen as the molecules to be engulfed). We remark that in our endosome rules, the objects inside and on the surface of the created endosome are explicitly defined. This is different from the approach of [5] in which in the case of pino rules the surface objects are randomly distributed to the two resulting membranes. More formally, the applicability of an endosome rule is possible in the following general case: let $(j, i) \in \mu$ and let $endo_E(u, v)$ be a rule belonging to the surface of the membrane i, then it can be applied only if u is a submultiset of the objects contained on the surface of the membrane i, and only if v is a submultiset of the objects contained inside the membrane j. The result of the application of such a rule is the creation of an endosome inside membrane i containing u on its surface and containing v inside. The endosome itself behaves like a membrane having on its surface rules E.

We can now formally define a PE System as follows.

Definition 1. A *PE System* Π is a tuple $(V, \mu, w_1, \ldots, w_n, z_1, \ldots, z_n, R_1, \ldots, R_n)$ where:

- V is an *alphabet* whose elements are called *objects*;

- $\mu \subset \mathbb{N} \times \mathbb{N}$ is a *membrane structure*;

- w_i with $1 \leq i \leq n$ are strings from V^* representing multisets over V associated with the content of membranes $1, 2, \ldots, n$ of μ;

- z_i with $1 \leq i \leq n$ are strings from V^* representing multisets over V associated with the surfaces of membranes $1, 2, \ldots, n$ of μ;

- R_i with $1 \leq i \leq n$ are finite sets of *evolution* and *endosome* rules associated with the surfaces of the membranes $1, 2, \ldots, n$ of μ.

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

3 Universality of PE Systems

In this section we prove a universality result for PE Systems by showing that any matrix grammar with appearance checking can be simulated by a PE System. Before giving the result and its proof, we recall from [11] the definition of this 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 $(A_1 \rightarrow x_1, \ldots, A_n \rightarrow x_n)$ 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 $(A_1 \rightarrow x_1, \ldots, A_n \rightarrow x_n)$ 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 $A_i \rightarrow x_i$ 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_{ac}^λ, 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 $F = \emptyset$) is denoted by MAT^λ, when λ–rules are allowed, and by MAT, when such rules are not allowed. It is known that (1) $MAT \subset MAT_{ac} \subset CS$; (2) $MAT^\lambda \subset MAT_{ac}^\lambda = RE$, 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, $ac(G) \leq 2$ and the matrices in M are in one of the following forms:

1. $(S \rightarrow XA)$, with $X \in N_1, A \in N_2$;

2. $(X \rightarrow Y, A \rightarrow x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$;

3. $(X \rightarrow Y, A \rightarrow \#)$, with $X, Y \in N_1, A \in N_2$;

4. $(X \rightarrow \lambda, A \rightarrow x)$, with $X \in N_1, A \in N_2, x \in T^*, |x| \leq 2$.

Moreover, there is only one matrix of type 1, and F consists exactly of all rules $A \rightarrow \#$ appearing in matrices of type 3. 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 (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 $m_0 : (S \rightarrow X_{init} A_{init})$ 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, $m_i : (X \rightarrow \alpha, A \rightarrow x)$, 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, $m_i : (X \rightarrow Y, A \rightarrow \#)$, with $k+1 \leq i \leq n, X, Y \in N_1$, and $A \in N_2$.

Since the grammar is in strong binary normal form, we have (at most) two symbols $B^{(1)}$ and $B^{(2)}$ in N_2 such that the rules $B^{(j)} \rightarrow \#$ appear in matrices m_i with $k+1 \leq i \leq n$.

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 PE Systems are universal by showing that the family, denoted $PsE_2(ncoo)$, of sets $Ps(\Pi_e)$ of results computed by PE Systems with at least two 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 endosomes.

Theorem 2. $PsE_2(ncoo) = PsRE$.

Proof. It is enough to show that for a matrix grammar G in strong binary normal form there is a PE System Π_G such that $Ps(\Pi_G) = \Psi_T(L(G))$. We assume that the output of this PE System is given by the objects sent out from the skin membrane. The alphabet of objects V we take into consideration is given by $T \cup N_1 \cup N_2 \cup \{c\} \cup \{c_i, d_i, d_i' \mid i = 1, 2\}$. We build Π_G as a system with a root membrane, labeled 1, and one child membrane labeled 2, namely $\mu = \{(1, 2)\}$. All the objects encoding the grammar will be stored inside membrane 1 and the matrices will be simulated by membrane 2. The initial configuration is given by the objects corresponding to X_{init} and A_{init} contained in membrane 1, namely objects of w_1, and by the token c contained on the surface of membrane 2, namely $z_2 = \{c\}$. Differently, w_2 and z_1 are initially empty multisets.

This PE System works as follows: it has a cyclic behaviour such that, at the beginning of the cycle, at most one endosome in membrane 2 can be created and, if possible, all terminal symbols inside membrane 1 are sent out as output symbols. The created endosome can start a series of steps resulting in the interpretation of the application of a matrix or, differently, it can start a checking phase to model the fact that, if there exist non–terminal symbols which cannot be rewritten by any grammar, then the computation will not halt. In the case in which Π_G starts the simulation of a matrix of type 2 or 4 (a matrix m_i with $1 \leq i \leq k$), the involved non–terminals are taken by the endosome which contains as rules the ones interpreting the matrix. Objects will be sent into membrane 2 by these rules creating the result of the

application of the the corresponding matrix to the non–terminals. Subsequently, these objects are sent out to membrane 1 to restart the cyclic behaviour. We recall that during this process no other endosomes can be created, hence no other matrices can be simulated. Differently, in the case in which a matrix of type 3 (a matrix m_i with $k+1 \leq i \leq n$) is applied, the single non–terminal of N_1 is taken into the endosome. The endosome will work in the same sense of the endosomes interpreting matrices of type 2 and 4 even though, at the end of the application of this matrix, instead of restarting with the cyclic behaviour, a checking process is started. This process checks, by creating endosomes, the presence of the proper non–terminal symbol $B^{(j)}$. If this symbol is found, a special endosome is created which will introduce a trap symbol in this PE System so that the computation will not halt. Analogously, if the symbol is not found, an endosome will restore the configuration of this PE System so that the cyclic behaviour can start again.

   We list now the rules of Π_G. Membrane 1 contains just one single set of rules to create the output of the PE System:

1. $\{(a,in) \rightarrow (a,out) \mid \forall a \in T\}$. All terminal objects in membrane 1 are sent out as output.

The simulation of any matrix is done by the rules of membrane 2 which are the following:

1. $\forall X \in N_1 \cup N_2$. $endo_{(X,in)\rightarrow(\#,out)}(c,X)$. If any non–terminal is present in membrane 1, Π_G will always be able to create, by using an endosome, a trap symbol inside membrane 2. This will ensure that, if a derivation of G reaches a deadlock configuration, then Π_G can always enter an endless configuration.

2. $\forall a \in N_1 \cup N_2 \cup T$. $(a,in) \rightarrow (a,out)$. Every terminal and non–terminal present inside membrane 2 is sent out to membrane 1.

3. $(c,in) \rightarrow (c,here)$. Object c inside membrane 2 is restored on the surface of membrane 2 so that other endosomes can be created.

4. $(\#,in) \rightarrow (\#,in)$. The trap symbol lets this computation not to be recognized such.

5. $m_i : (X \rightarrow \alpha, A \rightarrow x), 1 \leq i \leq k$. $endo_{(X,in)\rightarrow(\alpha,out),(A,in)\rightarrow(x,out),(c,here)\rightarrow(c,out)}(c,XA)$. For any rule of type 2 and 4, we create an endosome by taking XA from membrane 1 and c from the surface of membrane 2 (this locks the creation of other endosomes). The endosome contains rules to rewrite X and A with the result of applying the matrix. Object c is not consumed and sent out to membrane 2 together with α and x.

6. $m_i : (X \rightarrow Y, A \rightarrow \#), k+1 \leq i \leq n$. $endo_{(X,in)\rightarrow(Y,out),(c,here)\rightarrow(c_i,out)}(c,X)$. For any rule with appearance checking, we create an endosome by taking only X from membrane 1 and c from the surface of membrane 2 (this locks the creation of other endosomes). The endosome contains rules to rewrite X with Y and c with c_i. Both objects are sent out to membrane 2.

7. $(c_i,in) \rightarrow (c_i,here)(d_i,here)$. Object c_i, together with a new object d_i, is moved on the surface of membrane 2.

8. $endo_{(B^{(i)},in)\rightarrow(\#,out)}(c_i,B^{(i)})$. This implements the appearance checking feature of grammar G. We create, if possible, an endosome by taking only $B^{(i)}$ from membrane 1 and c_i from the surface of membrane 2. The endosome creates a trap symbol in membrane 2; this will make Π_G start an endless computation.

9. $(d_i,here) \rightarrow (d_i',here)$. The symbol d_i is rewritten in the same place as d_i'. This is done even if also rule 8 can be applied. However, in the case that rule 8 cannot be applied (namely $B^{(i)}$ was not present), this completes the appearance checking operation and lets Π_G start an operation which will restart its cyclic behaviour.

10. $endo_{(c_i,here)\rightarrow(c,out),(d_i',here)\rightarrow\lambda}(c_id_i',\emptyset)$. This endosome lets Π_G restart its cyclic behaviour. We create an endosome by simply taking both the control symbols only c_1 and d_i' from the surface of membrane 2. The endosome destroys d_i' and rewrites c_i with c in membrane 2 (restarting Π_G will be obtained by applying rule 3).

It is clear that these rules, applied in a proper order, provide the correct interpretation of the application of any matrix to the starting symbols of the grammar and, consequently, we get $Ps(\Pi_G) = \Psi_T(L(G))$ which concludes the proof. $\square$

4   An Application: the EGF Signaling Pathway

In this section we give an application of PE systems to the description of the initial phases of the EGFR signalling cascade.
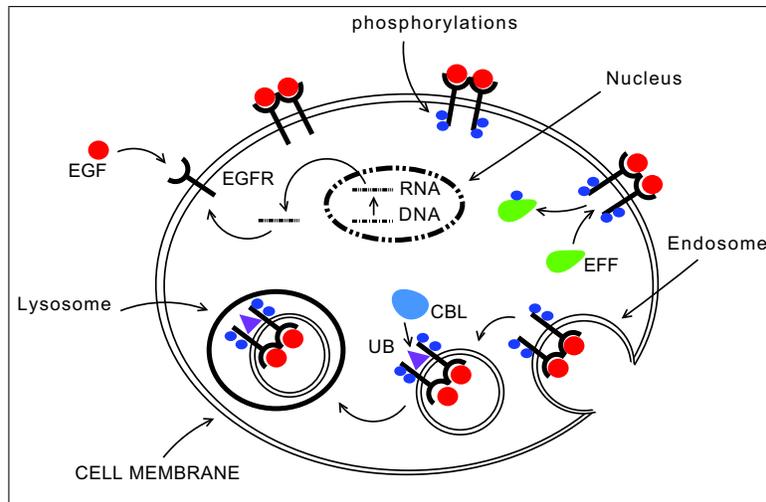
Figure 2: The EGF signaling pathway (picture taken from [3]).

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 ligand 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. Epidermal growth factor receptors ($EGFR$) 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 (COM). Upon activation, EGFR undergoes a transition from a monomeric form to an active dimeric one (DIM). EGFR dimerization stimulates its intracellular phosphorylation ($DIMp$) 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 (UB) to the dimer (ubiquitination). The ubiquitin protein targets the dimers for lysosomal degradation (see Figure 2).

The PE system modeling the EGF is given in Figure 3. Membrane 1 models the environment external
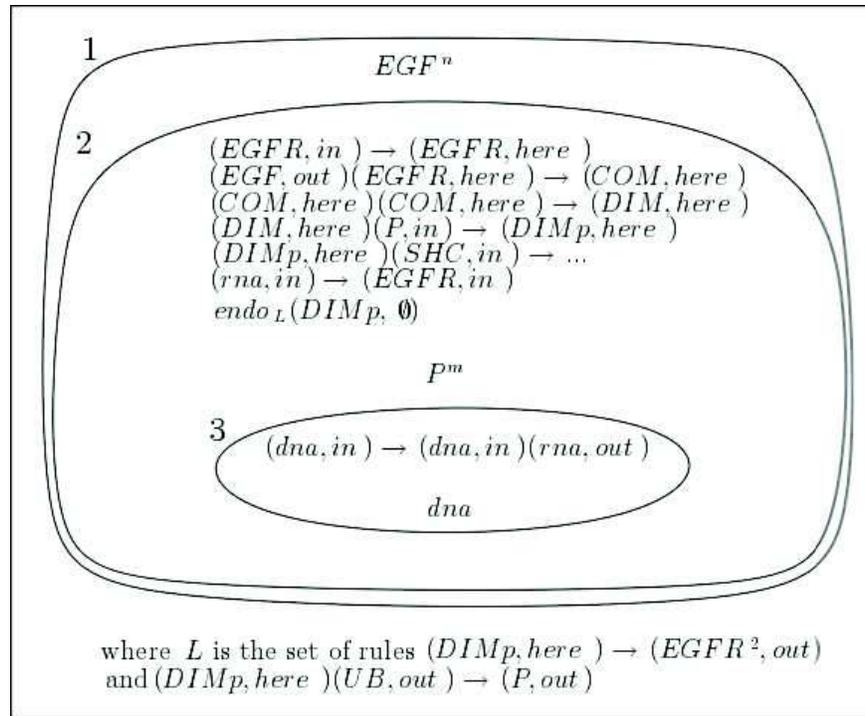
Figure 3: A PE systems model of the EGF signaling pathway. The rules are represented inside the membranes.

to the cell, membrane 2 represents the cell surface and membrane 3 is the nucleus. 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 move on the surface of the membrane. The complex of EGF with the receptor is obtained by rewriting EGF and $EGFR$ with the complex COM on the surface of membrane 2. After the binding of two complexes we can bind them leading to a dimer DIM. Such a dimer, present on the surface of the membrane, can be phosphorylated by a phosphorus P inside the cell. Such phosphorilated dimer $DIMp$ could interact with protein SHC and start a chain of interactions we do not model here aimed at activating cell proliferation. Furthermore, it can be enclosed in an endosome which could either decompose the $DIMp$ dimer into its original components (in order to recycle the two $EGFR$ proteins) or, if ubiquitine UB is present, degradate the $DIMp$ dimer and release the phosporus. The nucleus of the cell (membrane 3) is responsible for the production of $EGFR$ through the DNA and RNA (dna and rna). The rna reaches the cell cytoplasm and there it produces $EGFR$ which is sent, again, to the cell surface.

5 Future Work and Conclusions

In this paper we have presented an extension of P Systems, called P Systems with Endosomes (PE Systems), in which endosomes can be explicitly modeled. PE Systems uses some ideas taken from other variants of P Systems, in particular as regards objects which can be stored on the surface of the membranes we got inspiration by P Systems with peripheral proteins [6, 13] and by membrane systems with surface objects [1, 2]. Furthermore, as regards other calculi, operations for modeling transportation mechanisms have already been introduced in Brane Calculi [4] and in P Systems with transport and embedded proteins [9]. Although similar, PE Systems permit to model in a clearer way these mechanisms. An analysis of PE Systems and Brane Calculi [4] (and also some of their variants like projective Brane

Calculi [7]) could be done along the line of the one done in [5, 12] for P Systems and Brane Calculi.

As regards expressiveness of this formalism, we have shown that PE Systems are universal even if only the simplest form of evolution rules is considered, namely non–cooperative rules. This expressiveness is achieved by the use of endosomes as classical P Systems with this kind of rules are shown not to be universal [10].

At the end of the paper we have given an application example describing the modeling of the initial phases of the EGFR signalling cascade.

Bibliography

[1] Aman, B., Ciobanu, G.: Membrane Systems With Surface Objects. Proceedings of the Int. Workshop on Computing with Biomolecules (CBM 2008), Vienna, 17–29, 2008.

[2] Aman, B., Ciobanu, G.: Mutual Mobile Membrane Systems With Objects on Surface. Proceedings of the Seventh Brainstorming Week on Membrane Computing (BWMC09), Seville, 2009.

[3] Barbuti, R., Maggiolo–Schettini, A., Milazzo, P. Tini, S.: P Systems with Transport and Diffusion Membrane Channels. Int. Workshop on Concurrency, Specification and Programming (CS&P'08), Gross Vaeter, Germany, September, 2008.

[4] Cardelli, L.: Brane calculi. Interactions of biological membranes. In: Danos, V., Schachter, V. (Eds.), LNCS **3082** (2005), pp. 257-280.

[5] Cardelli, L., Păun, G.: An universality result for a (mem)brane calculus based on mate/drip operations. Internat. J. Found. Comput. Sci. **17**(1), pp. 49–68.

[6] Cavaliere, M., Seawards, S.: Membrane systems with peripheral proteins: transport and evolution. Proc. of the First Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2006), ENTCS **171** (2007), pp. 37–53.

[7] Danos, V. Pradalier, S.: Projective Brane Calculus. Proc. of the Fourth Conference on Computational Methods in Systems Biology (CMSB04), LNCS **3082** (2005), pp. 134–148.

[8] Freund, R., Oswald, M.: P systems with activated/prohibited membrane channels. Proc. of WMC 2002, LNCS **2597** (2003), pp. 261–269.

[9] Krishna, S.N.: Membrane computing with transport and embedded proteins. Theoretical Computer Science **410** (2009), pp. 355–375.

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

[11] Păun, G.: Membrane Computing. An Introduction. Springer (2002).

[12] Păun, G.: Membrane computing and brane calculi. Old, new, and future bridges. Theoretical Computer Science **404**(1-2), pp. 19–25.

[13] Păun, A., Popa, B.: P Systems with Proteins on Membranes. Fundamenta Informaticae **72**(4) (2006), pp. 467 – 483.

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

Roberto Barbuti was born in 1953. He received a Laurea degree in Computer Science in 1977 from the University of Pisa. In 1982 he became Assistant Professor at University of Pisa. He took the position of Associate Professor in 1989, and the position of Full Professor in 2000. Presently, he is working in the research fields of Formal Methods for Systems Biology.

Giulio Caravagna was born in 1982. He received a Bachelor and a Master Degree in Computer Science from the Univeristy of Pisa in years 2005 and 2007, respectively. At present, he is a PhD student in Computer Science working in the fields of Formal Methods for Systems Biology and Natural Computing.

Andrea Maggiolo Schettini was born in 1938. He received a laurea degree in Physics from the University of Genova and from 1966 to 1968 he was a researcher of the Italian National Institute for Nuclear Physics in Bologna. After shifting his interest to fundamental studies in computer science, from 1968 to 1981 he was a researcher of the National Research Council in Naples and in Pisa. From 1981 to 1983 he has been full professor of Computer Science at the University of Turin and since 1983 he is full professor of Computer Science at the University of Pisa. He has done research in computability theory, semantics of programming languages, specification and verification of concurrent and distributed systems. His present interests include Systems Biology and Natural Computing.

Paolo Milazzo was born in 1979. He received a Master Degree in Computer Science in 2003 at the University of Bologna and a Ph.D. in Computer Science in 2007 at the University of Pisa. Actually, he is research fellow at the Department of Computer Science of the University of Pisa. working on Computational Systems Biology and Natural Computing with a particular focus on the definition and application of Formal Methods.