Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 3, pp. 268-279

Adding Lifetime to Objects and Membranes in P Systems

B. Aman, G. Ciobanu

Bogdan Aman, Gabriel Ciobanu
Romanian Academy, Institute of Computer Science
and A.I.Cuza University of Iaşi, Romania
E-mail: baman@iit.tuiasi.ro, gabriel@info.uaic.ro

Abstract: Membrane systems are computing devices inspired from the cell functioning. A feature of membrane systems is the fact that objects and membranes are persistent. In fact, this is not quite true in the real world: cells and intracellular proteins have a well-defined lifetime. Inspired from these biological facts, we define a model of membrane systems in which each membrane and each object has attached a lifetime. Some results show that this model is at least as powerful as the usual one.

1 Introduction to Membrane Computing

Membrane systems are essentially parallel and nondeterministic computing models inspired by the compartments of eukaryotic cells and their biochemical reactions. The structure of the cell is represented by a set of hierarchically embedded regions, each one delimited by a surrounding boundary (called membrane), and all of them contained inside an external special membrane called *skin*. The molecular species (ions, proteins, etc.) floating inside cellular compartments are represented by multisets of objects described by means of symbols or strings over a given alphabet. The objects can be modified or communicated between adjacent compartments. Chemical reactions are represented by evolution rules which operate on the objects, as well as on the compartmentalized structure (by dissolving, dividing, creating, or moving membranes).

A membrane system can perform computations in the following way: starting from an initial configuration which is defined by the multiset of objects initially placed inside the membranes, the system evolves by applying the evolution rules of each membrane in a nondeterministic and maximally parallel manner. A rule is applicable when all the objects which appear in its left hand side are available in the region where the rule is placed. The maximally parallel way of using the rules means that in each step, in each region of the system, a multiset of rules is chosen which is maximal and applicable, namely a multiset of rules such that no further rule can be added to the multiset. A halting configuration is reached when no rule is applicable. The result is represented by the number of objects from a specified membrane.

Several variants of membrane systems are inspired by different aspects of living cells (symport and antiport-based communication through membranes, catalytic objects, membrane charge, etc.). Their computing power and efficiency have been investigated using the approaches of formal languages, grammars, register machines and complexity theory. Membrane systems (also called P systems) are presented together with many variants and examples in [21]. Several applications of these systems are presented in [12]. An updated bibliography can be found at the P systems web page [22].

For an alphabet $V = \{a_1, \ldots, a_n\}$, we denote by V^* the set of all strings over V; λ denotes the empty string and $V^+ = V^* \backslash \{\lambda\}$. A multiset over V is represented by a string over V (together with all its permutations), and each string precisely identifies a multiset.

A *language (over V)* is any subset of V^*; a language is denoted usually by L. Given a language L, we define the set $Ps(L) = \{\Psi_V(x) \mid x \in L\}$ called the *Parikh image of L*. If FL is a family of languages, then $PsFL$ denotes the family of Parikh images of languages in FL.

Definition 1. A P system of degree $n \geq 1$ is a construct

$$\Pi = (V, T, C, H, \mu, w_1, \ldots, w_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), i_O)$$

where:

1. V is an alphabet of symbols; its elements are called objects;
2. $T \subseteq V$ is the terminal (or output) alphabet;
3. $C \subseteq V$, $C \cap T = \emptyset$ is the alphabet of catalysts;
4. H is a set of membrane labels;
5. $\mu \subseteq H \times H$ is a tree that describes the membrane structure, such that $(i, j) \in \mu$ denotes that the membrane labelled by j is contained in the membrane labelled by i;
6. $w_i \in V^*$, for each $1 \leq i \leq n$, is a multiset of objects initially assigned to membrane i;
7. R_i, for all $1 \leq i \leq n$, is a finite set of evolution rules that is associated with membrane i; an evolution rule is a multiset rewriting rule of the form $u \rightarrow v$, with $u \in V^+$, either $v = v'$ or $v = v'\delta$, $v' \in ((V \times \{here, out\}) \cup (V \times \{in_j \mid 1 \leq j \leq n\}))^*$, and δ a special symbol not appearing in V;
8. ρ_i, for all $1 \leq i \leq n$, is a partial order relationship defined over the rules in R_i specifying a priority relation between these rules;
9. i_O is the label of an elementary membrane of μ that identifies the output region.

Therefore, a P systems of degree $n \geq 1$ consists of a membrane structure μ containing $n \geq 1$ membranes where each membrane i gets assigned a finite multiset of objects w_i and a finite set of evolution rules R_i. An evolution rule is a multiset rewriting rule which consumes a multiset of objects from V and produces a multiset of pairs (a,t), with $a \in V$ and $t \in \{here, out\} \cup \{in_j \mid 1 \leq j \leq n\}$ a *target* specifying where to move the objects after the application of the rule. As well as this, an evolution rule can produce the special object δ to specify that, after the application of the rule, the membrane containing the special object δ has to be dissolved. After dissolving a membrane, all objects and membranes previously present in it become elements of the immediately upper membrane, while the rules of the dissolved membrane are removed.

We use *P systems without lifetimes* instead of *P systems* in order to make a clear distinction from the *P systems with lifetimes* which are introduced in what follows.

2 P Systems with Lifetimes

The evolution of complicated real systems frequently involves various interdependence among components. Some mathematical models of such systems combine both discrete and continuous evolutions on multiple time scales with many orders of magnitude. For example, in nature the molecular operations of a living cell can be thought of as such a dynamical system. The molecular operations happen on time scales ranging from 10^{-15} to 10^4 seconds, and proceed in ways which are dependent on populations of molecules ranging in size from as few as approximately 10^1 to approximately as many as 10^{20}. Molecular biologists have used formalisms developed in computer science (e.g. hybrid Petri nets) to get simplified models of portions of these transcription and gene regulation processes. According to molecular cell biology [18]:

(i) "the life span of intracellular proteins varies from as short as a few minutes for mitotic cycles, which help regulate passage through mitosis, to as long as the age of an organism for proteins in the lens of the eye."

(ii) "Most cells in multicellular organisms ... carry out a specific set of functions over periods of days to months or even the lifetime of the organism (nerve cells, for example)."

It is obvious that lifetimes play an important role in the biological evolution. We use an example from the immune system.

Example 1 ([18]). T-cell precursors arriving in the thymus from the bone marrow spend up to a week differentiating there before they enter a phase of intense proliferation. In a young adult mouse the thymus contains around 10^8 to 2×10^8 thymocytes. About 5×10^7 new cells are generated each day; however, only about 10^6 to 2×10^6 (roughly $2-4\%$) of these will leave the thymus each day as mature T cells. Despite the disparity between the numbers of T cells generated daily in the thymus and the number leaving, the thymus does not continue to grow in size or cell number. This is because approximately 98% of the thymocytes which develop in the thymus also die within the thymus.

Inspired from these biological facts, we add lifetimes to objects and membranes. We use a global clock to simulate the passage of time in a membrane system.

Definition 2. A P system with lifetimes of degree $n \geq 1$ is a construct

$$\Pi = (V_t, T, C, H_t, \mu_t, w_1, \ldots, w_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), i_O)$$

where:

1. $V_t = V \times (\mathbb{N} \cup \infty)$ is a set of pairs of the form (a, t_a), where $a \in V$ is an object (as in Definition 1) and $t_a \in (\mathbb{N} \cup \infty)$ is the lifetime of the object a;
2. T, C are as in Definition 1;
3. $H_t = H \times (\mathbb{N} \cup \infty)$ is a set of set of pairs of the form (h, t_h), where $a \in H$ is a membrane label (as in Definition 1) and $t_h \in (\mathbb{N} \cup \infty)$ is the lifetime of the membrane h;
4. $\mu_t \subseteq H_t \times H_t$ is a tree that describes the membrane structure, such that $((i,t_i),(j,t_j)) \in \mu_t$ denotes that the membrane labelled by j, with the lifetime j_t, is contained in the membrane labelled by i, with the lifetime i_t;
5. $w_i \subseteq (V_t)^*$ is a multiset of pairs from V_t assigned initially to membrane i;
6. R_i, for all $1 \leq i \leq n$, is a finite set of evolution rules that is associated with membrane i of the following forms:
   (a) $u \to v$, with $u \in V_t^+$, either $v = v'$ or $v = v'\delta$, $v' \in ((V_t \times \{here, out\}) \cup (V_t \times \{in_j \mid 1 \leq j \leq n\}))^*$; δ is a special symbol not appearing in V;
   (b) $(a,t) \to (a, t-1)$, for all $a \in V$ and $t > 0$
   If an object a is not involved in a rule of type (a) and it has a lifetime $t > 0$, then its lifetime is decreased.
   (c) $(a, 0) \to \lambda$, for all $a \in V$
   If an object a has the lifetime 0 then the object is replaced with the empty multiset λ, thus simulating the degradation of proteins.
   (d) $[\,]_{(i,t)} \to [\,]_{(i,t-1)}$, for all $1 \leq i \leq n$
   In each evolution step the lifetime of each membrane of the membrane structure is decreased with one.
   (e) $[\,]_{(i,0)} \to [\delta]_{(i,0)}$, for all $1 \leq i \leq n$
   If the lifetime of a membrane reaches 0 the membrane is dissolved.
7. ρ_i, for all $1 \leq i \leq n$, is a partial order relationship defined over the rules in R_i specifying a priority relation between these rules;

8. i_O is the label of an elementary membrane of μ_t that identifies the output region.

These rules are applied according to the following principles:

1. All the rules are applied in parallel: in a step, the rules are applied to all objects and to all membranes; an object can be used only by one rule, non-deterministically chosen (there is no priority among rules), but any object which can evolve by a rule of any form, should evolve.

2. If a membrane is dissolved, then all the objects in its region are left free in the region immediately above it. Because all rules are associated with membranes, the rules of a dissolved membrane are no longer available at the next step.

3. The skin membrane has the lifetime equal to ∞, so it can never be dissolved.

4. If a membrane or object has the lifetime equal to ∞, when applying the rules simulating the passage of time we use the equality $\infty - 1 = \infty$.

The computation stops when the membrane system contains only objects and membranes that have the lifetime equal to ∞.

Example 2. The concentration of a molecule can be adjusted quickly only if the lifetime of the molecule is short [1]. It is natural to think of signaling systems in terms of the changes produced when a signal is delivered. But it is just as important to consider what happens when a signal is withdrawn. During development transient signals often produce lasting effects: they can trigger a change that persists indefinitely, through cell memory mechanisms. But in most cases, especially in adult tissues, when a signal ceases, the response fades. The signal acts on a system of molecules that is undergoing continual turnover, and when the signal is shut off, the replacement of the old molecules by new ones wipes out the traces of its action. It follows that the speed of reaction to shutting off the signal depends on the rate of turnover of the molecules that the signal affects. It may not be as obvious that this turnover rate also determines the promptness of the response when the signal is turned on.

The Figure 1 shows the predicted relative rates of change in the intracellular concentrations of molecules with differing turnover times when their rates of synthesis are increased suddenly by a factor of 10. The concentrations of those molecules that are normally being rapidly degraded in the cell (red lines) change quickly, whereas the concentrations of those that are normally being slowly degraded (green lines) change proportionally more slowly. The numbers (in blue) on the right-hand side are the half-lives assumed for each of the different molecules.

Consider, for example, two intracellular molecules X and Y, both of which are normally maintained at a concentration of 1000 molecules per cell. Molecule X has a slow turnover rate: it is synthesized and degraded at a rate of 10 molecules per second, so that each molecule has an average lifetime in the cell of 100 seconds. Molecule Y turns over 10 times as quickly: it is synthesized and degraded at a rate of 100 molecules per second, with each molecule having an average lifetime of 10 seconds. If a signal acting on the cell boosts the rates of synthesis of both X and Y tenfold without any change in the molecular lifetimes, at the end of 1 second the concentration of Y will have increased by nearly 900 molecules per cell $(10 \times 100 - 100)$ while the concentration of X will have increased by only 90 molecules per cell. In fact, after its synthesis rate has been either increased or decreased abruptly, the time required for a molecule to shift halfway from its old to its new equilibrium concentration is equal to its normal half-life - that is, it is equal to the time that would be required for its concentration to fall by half if all synthesis were stopped (Figure 1).

The same principles apply to proteins as well as to small molecules and to molecules in the extracellular space as well as to those in cells. Many intracellular proteins that are rapidly degraded have short half-lives, some surviving less than 10 minutes; in most cases these are proteins with key regulatory roles,

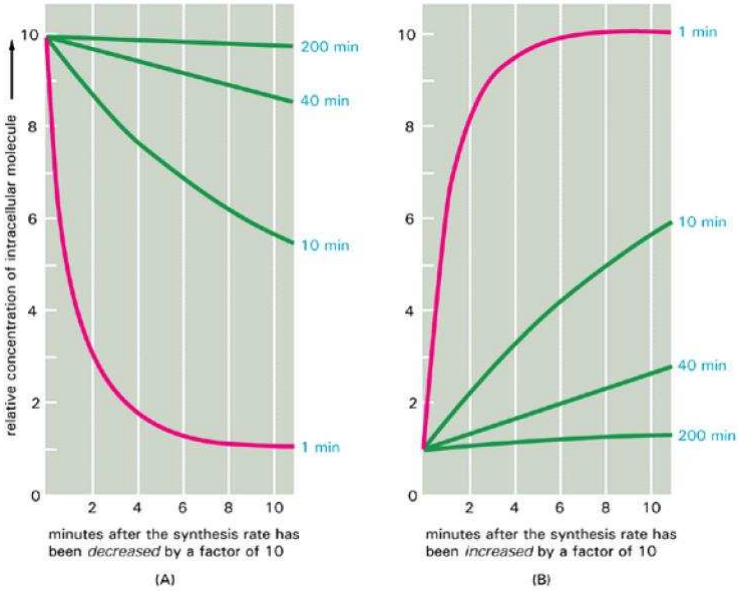


Figure 1: The importance of rapid turnover [1]

whose concentrations are rapidly regulated in the cell by changes in their rates of synthesis. Likewise, any covalent modifications of proteins that occur as part of a rapid signaling process - most commonly the addition of a phosphate group to an amino acid side chain - must be continuously removed at a rapid rate to make such signaling possible.

Example 3. The scenario presented in Example 2 can be modelled using P systems with lifetimes. Consider the membrane configuration

$$[(X_s,\infty)(X,0)^{10}\dots(X,99)^{10}(Y_s,\infty)(Y,0)^{100}\dots(Y,9)^{100}]_{(cell,\infty)}$$

describing the structure of the cell when in equilibrium. X_s and Y_s represent the molecules from which X and Y are synthesized, while $(X,t_1)^{10}$ and $(Y,t_2)^{100}$ represent the fact that there are 10 molecules X which have the lifetime equal to t_1 and 100 molecules Y which have the lifetime equal to t_2. The rules describing the evolution of this system are:

1. $(X,0)\to\lambda$

   A molecule X which is at the end of its lifetime is degraded.

2. $(Y,0)\to\lambda$

   A molecule Y which is at the end of its lifetime is degraded.

3. $(X_s,\infty)\to(X_s,\infty)(X,99)^{10}$

   Each second 10 new X molecules are synthesized.

4. $(Y_s,\infty)\to(Y_s,\infty)(Y,9)^{100}$

   Each second 100 new Y molecules are synthesized.

5. $(X,t)\to(X,t-1), t>0$

   The lifetime of a molecule X which is not involved in any reaction is decreased.

6. $(Y,t)\to(Y,t-1), t>0$

   The lifetime of a molecule Y which is not involved in any reaction is decreased.

After applying rules in a maximal manner after each second we reach the initial configuration $[(X_s,\infty)(X,0)^{10}\dots(X,99)^{10}(Y_s,\infty)(Y,0)^{100}\dots(Y,9)^{100}]_{(cell,\infty)}$.

If two signals enter the cell (we model the signals by the pairs (c_X,t_c) and (c_Y,t_c)) then we consider two new rules:

7. $(c_X,t_c)(X_s,\infty)\rightarrow(c_X,t_c-1)(X_s,\infty)(X,99)^{100}$

   If an object c_X is present in the cell then the rate of synthesis of X is increases 10 times.

8. $(c_Y,t_c)(Y_s,\infty)\rightarrow(c_Y,t_c-1)(Y_s,\infty)(Y,9)^{1000}$

   If an object c_Y is present in the cell then the rate of synthesis of Y is increases 10 times.

When adding this rules we also add some priorities between rules, namely $7>5$ and $8>6$. After one second the concentration of Y increases by nearly 900 molecules per cell $(10\times100-100)$ while the concentration of X increases by only 90 molecules per cell.

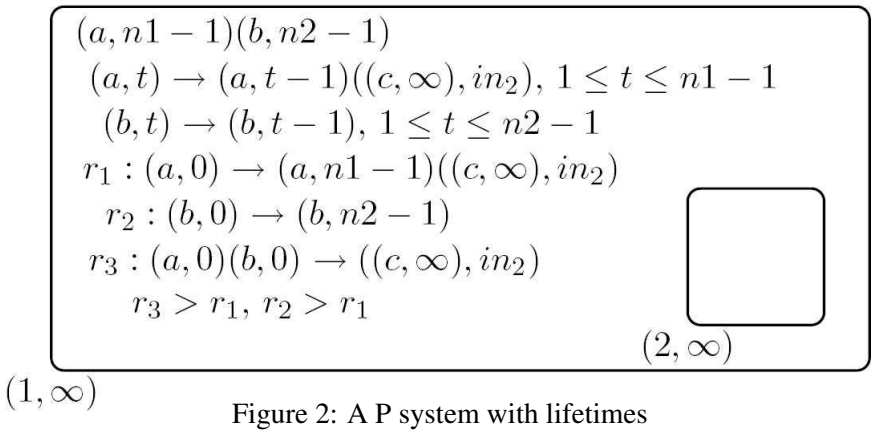


Figure 2: A P system with lifetimes

In Figure 2 an example of a P system with lifetimes is shown. Graphically, the boxes represent the membranes and their nesting reflects the hierarchy. Membrane 1 represent the skin membrane. Both membranes have a lifetime equal to ∞ meaning that no dissolving rule is not necessary. Inside membrane 1 we have the initial multiset of pairs, the evolution rules and some priorities between them. In the evolution rule we omit the subscript *here* for objects, in the products, that remain in the same membrane.

The defined P system with lifetimes computes the least common multiple of n_1 and n_2, namely $lcm(n_1,n_2)$. The idea is to put a pair $(a,n1-1)$ and a pair $(b,n2-1)$ at the beginning of the computation, and to produce a pair (c,∞) (which is send in membrane 2) each time the lifetime of the object a is decreased. The first time the objects a and b appear together with the lifetime 0 is exactly after $lcm(n_1,n_2)$ time units. At this moment in membrane 2 are $lcm(n_1,n_2)$ objects c, which represent the output of the system.

3 Systems with and without Lifetimes

The following results describe certain relationships between P systems with lifetimes and P systems without lifetimes, and between similar P systems with lifetimes.

Proposition 4. For every P system without lifetimes there exists a P system with lifetimes providing the same output by performing an equal number of steps.

Proof: [Proof (Sketch)] It is easy to state that the class of P systems with lifetimes includes the class of P systems without lifetimes, since we can assign ∞ to all lifetimes appearing in the membrane structure and evolution rules. □

A somehow surprising result is that P systems with lifetimes can be simulated by P systems without lifetimes.

Proposition 5. For every P system with lifetimes there exists a P system without lifetimes providing the same output by performing an equal number of steps.

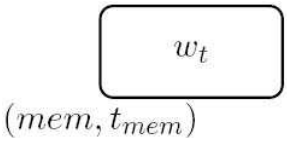
Proof: We use the notation $rhs(r)$ to denote the multisets of pairs which appear in the right hand side of a rule r. This notation is extended naturally to multisets of rules: given a multiset of rules R, the right hand side of the multiset $rhs(R)$ is obtained by adding the right hand sides of the rules in the multiset, considered with their multiplicities.

Each object $a \in V$ from a P system with lifetimes has a maximum lifetime (we denote it by $lifetime(a)$), which can be calculated as follows:

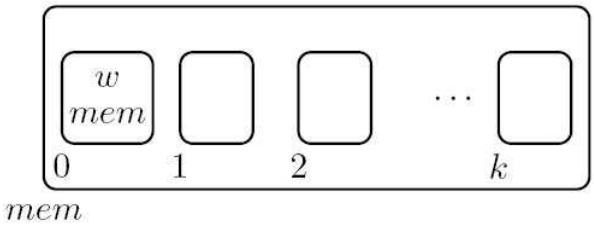
$lifetime(a) = max(\{t \mid (a,t) \in w_i, 1 \leq i \leq n\} \cup \{t \mid (a,t) \in rhs(R_i), 1 \leq i \leq n\})$

In what follows we present the steps which are required to build a P system without lifetimes starting from a P system with lifetimes, such that both provide the same output.

1. A membrane structure from a P system with lifetimes



is translated into a membrane structure of a P system without lifetimes in the following way



The lifetimes of elements from a P system with lifetimes are simulated using the membranes 0, ..., k in the corresponding P system without lifetimes as we show at the next steps of the translation. The value of k is the maximum from the finite lifetime of objects and surrounding membrane *mem*, namely $k = max(\{lifetime(a) \mid a \in V, lifetime(a) \neq \infty\} \cup \{t_{mem} \mid t_{mem} \neq \infty\})$. If an object or the surrounding membrane has the lifetime equal to ∞ in the P system with lifetimes then we do not need to count the passage of time, namely to use the membranes 0, ..., k in the P system without lifetimes for the corresponding object or membrane. The object *mem* placed inside the membrane labelled 0 is used to simulate the passage of time for the membrane. The initial multiset of objects w_t from membrane *mem* in the P system with lifetimes, in translated into the multiset w which is added into membrane 0 inside membrane *mem* in the corresponding P system without lifetimes since all objects from the initial multiset are just starting their life.

2. The rules $(a,t) \rightarrow (a,t-1)$, for all $a \in V$, $t > 0$ and $t \neq \infty$ from the P system with lifetimes can be simulated in the P system without lifetimes using the following rules:
   (a) $a \rightarrow (ao_{ai}, out)$ placed inside membrane i, for all $0 \leq i \leq lifetime(a) - 1$
   The object o_{ai} is used to keep track of the units of time that have past from the lifetime of the object a

(b) $ao_{aj} \to (a, in_{j+1})$, placed inside membrane *mem*, for all $0 \leq j \leq lifetime(a) - 1$ and $a \in V$

This rules together with the previous ones simulate the passage of a unit of time from the lifetime of object a in the P system with lifetimes, by moving object a from a membrane j to a membrane $j+1$ for $0 \leq j \leq lifetime(a) - 1$ in the P system without lifetimes.

3. The rules $(a, 0) \to \lambda$, for all $a \in V$ from the P system with lifetimes can be simulated in the P system without lifetimes using the following rules:

(a) $a \to \lambda$ placed inside membrane $lifetime(a)$

If the object a reaches the membrane labelled with $lifetime(a)$ in the P system without lifetimes, it means that the lifetime of object a is 0 in the corresponding P system with lifetimes, so it is replaced by λ.

4. The rules $u_t \to v_t$ from the P system with lifetimes can be simulated in the P system without lifetimes using the following rules:

(a) $uo_{uj} \to (v, in_0)$

The multiset $o_u j$ contains objects of the form o_{aj}, where $a \in u$ and $0 \leq j \leq lifetime(a) - 1$. When replacing the multiset u with the multiset v, we also remove the objects o_{aj} that keep track of the life of the objects appearing in u since we do not need them anymore. We send the newly obtained multiset v in membrane 0 since all objects from this multiset are just starting their life.

5. The rules $[\,]_{(mem,t)} \to [\,]_{(mem,t-1)}$, for all $t > 0$ and $t \neq \infty$ from the P system with lifetimes can be simulated in the P system without lifetimes using the following rules:

(a) $mem \to (mem\, o_{mem\, i}, out)$ placed inside membrane i, for all $0 \leq i \leq lifetime(mem) - 1$

The object $o_{mem\, i}$ is used to keep track of the units of time that have past from the lifetime of the membrane *mem*;

(b) $mem\, o_{mem\, j} \to (mem, in_{j+1})$, placed inside membrane *mem*, for all $0 \leq j \leq lifetime(mem) - 1$

This rules together with the previous ones simulate the passage of a unit of time from the lifetime of membrane *mem* in the P system with lifetimes, by moving object *mem* from a membrane j to a membrane $j+1$ for $0 \leq j \leq lifetime(mem) - 1$ in the P system without lifetimes.

6. A rule $[\,]_{(mem,0)} \to [\delta]_{(mem,0)}$ from the P system with lifetimes can be simulated in the P system without lifetimes using the following rules:

(a) $mem \to (o_\delta, out)$ placed inside membrane $lifetime(mem) - 1$

If the object *mem* reaches the membrane labelled with $lifetime(mem) - 1$ in the P system without lifetimes, it means that the lifetime of membrane *mem* is 0 in the corresponding P system with lifetimes, so it needs to be dissolved.

(b) $o_\delta \to \delta(\delta, in_1) \ldots (\delta, in_k)$

Once this object is created by the previous rule, an object δ is created inside membrane *mem* and inside each membrane j, $0 \leq j \leq k$, is send an object δ. This means that all these membranes are dissolved, all the rules are deleted, and all objects are send in the parent membrane. The dissolving of the membranes take place after applying all other possible rules. At the moment of the dissolution the only existing objects are found in membrane *mem*. For each object a there exists an object o_{aj} that keeps track of the life of a, thus being able to continue the increment the life of a in the parent membrane.

The output membrane from the P system with lifetimes, is translated in the output membrane from the P system without lifetimes. After performing the same number of evolution steps in both systems, the output membranes contain the same multisets of objects. □

We are now able to prove the computational power of P systems with lifetimes. We denote by $\mathbb{N}tP_m(coo,tar)$ the family of sets of natural numbers generated by P systems with lifetimes of degree at most $m \geq 1$, using cooperative rules, and communication of objects through membranes. We also denote by $\mathbb{N}RE$ the family of all sets of natural numbers generated by arbitrary grammars.

Proposition 6. $\mathbb{N}tP_m(coo,tar) = \mathbb{N}RE$, for all $m \geq 1$.

Proof: [Proof (Sketch)] Since the outcome of each P system with lifetimes can be obtained by an P system without lifetimes, we cannot get more than the computability power of P systems. Therefore, according to Theorem 3.3.3 from [21], we have that the family $\mathbb{N}tP_m$of sets of natural numbers generated by P systems with lifetimes is the same as the family $\mathbb{N}RE$ of sets of natural number generated by arbitrary grammars. □

Remark 3.1. Consider the membrane system $\Pi = [[\,]_{(2,\infty)}(a,1)(a,1)]_{(1,\infty)}$ with the set of rules $R_1 = \{(a,1) \to (a,1)((c,\infty),in_2)^3; (a,1) \to (a,0); (a,0) \to \lambda\}$. Since the membranes have the lifetime ∞ it is not necessary to consider rules for decreasing lifetimes to membranes.

If we rewrite this as $\Pi' = [[\,]_{(2,\infty)}(a,1)(a,1)(d,t)]_{(1,\infty)}$, with $R_1 = \{(a,1) \to (a,1)((c,\infty),in_2)^3; (a,1) \to (a,0); (a,0) \to \lambda; (d,i) \to (d,i-1); (d,0) \to \lambda\}$ we have that after t units of time the membrane system Π has the same evolution as the membrane system Π'.

In automata theory the problem of optimizing a finite-state machine, meaning to find the machine with the minimum number of states that performs the same function, was addressed by the theorem of Myhill and Nerode; a fast algorithm doing this is the Hopcroft minimization algorithm [16].

Using a similar approach we want to optimize a given P system with lifetimes. This can be realized with the passage of time, namely all objects and membranes which are not used in rewriting rules and have a finite lifetime are eliminated.

Proposition 7. Let Π and Π' be two P systems with lifetimes such that:

1. $\Pi = (V_t, T, C, H_t, \mu_t, w_1, \ldots, w_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), i_O)$
2. $\Pi' = (V_t, T, C, H_t, \mu_t, w_1', \ldots, w_n', (R_1, \rho_1), (R_1', \rho_1'), \ldots, (R_n', \rho_n'), i_O')$
3. $i_O' = i_O$

   The output membrane is the same for the two membrane systems.
4. $w_i \subseteq w_i$, for all $1 \leq i \leq n$

   The initial multiset from membrane i of Π' contains the same objects as the initial multiset from membrane i of Π and some other objects together with their initial lifetimes.
5. $R_i' = R_i \cup \{(a,t) \to (a,t-1), (a,0) \to \lambda \mid a \in w_i' \backslash w_i\}$

   The set of rules R_i' contains all the rules of R_i and some additional rules to simulate the passage of time for all the objects appearing in $w_i' \backslash w_i$.
6. $\rho_i' = \rho_i$, for all $1 \leq i \leq n$

   The priority orders are the same for the two membrane systems.

Then the P systems with lifetimes Π and Π' have the same membrane structure and evolution after $max\{t \mid (a,t) \in w_i' \backslash w_i, 1 \leq i \leq n\}$ units of time.

Proof: [Proof (Sketch)] After $max\{t \mid (a,t) \in w_i' \backslash w_i, 1 \leq i \leq n\}$ units of time all objects which appeared in $w_i \backslash w_i'$ in the description of the membrane system Π' are consumed. In this case we have that the contents of the membranes of Π is the same as the contents of the membranes of Π', and the applicable rules for the two membrane systems are only the rules of R_i, $1 \leq i \leq n$. □

4 Related Work and Conclusion

We introduce a new class of P Systems, namely the P Systems with lifetimes. Lifetimes are assigned to each membrane and to each object. This new feature is inspired from biology where cells and intracellular proteins have a well defined lifetime. In order to simulate the passage of time, we use rules of the form $(a,t) \rightarrow (a,t-1)$ for objects, and $[\,]_{(i,t)} \rightarrow [\,]_{(i,t-1)}$ for membranes. If the lifetime of an object reaches 0 then the object is consumed by applying a rule of the form $(a,0) \rightarrow \lambda$, while if the lifetime of a membrane i reaches 0 then the membrane is marked for dissolution by applying a rule of the form $[\,]_{(i,0)} \rightarrow [\delta]_{(i,0)}$. After dissolving a membrane, all objects and membranes previously contained in it become elements of the immediately upper membrane.

We do not obtain a more powerful formalism by adding lifetimes to objects and to membranes into a P system. According to Proposition 4, Proposition 5 and Proposition 6, P systems with lifetimes and P systems without lifetimes have the same computational power. However the P systems with lifetimes are able to describe more naturally some biological phenomena involving timing, as in Example 3.

A similar idea appears in the framework of spiking P systems: considering a duration of life for spikes, but not for cells [15]. If a spike is not used in a number of steps larger than its lifetime, then it is removed.

There are also some papers using time in the context of membrane computing in a different manner than in this paper. In [8] a timed P system is introduced by associating to each rule a natural number representing the time of its execution. Then a P system which always produces the same result, independently from the execution times of the rules, is called a time-independent P systems. The notion of time-independent P systems tries to capture the class of systems which are robust against the environment influences over the execution time of the rules of the system. Other types of time-free systems are considered in [7, 9].

Time can also be used to "control" the computation, for instance by appropriate changes in the execution times of the rules during a computation, and this possibility has been considered in [11]. Moreover, timed P automata have been proposed and investigated in [5], where ideas from timed automata have been incorporated into timed P systems.

Frequency P systems has been introduced and investigated in [19]. In frequency P systems each membrane is clocked independently from the others, and each membrane operates at a certain frequency which could change during the execution. Dynamics of such systems have been investigated.

If one supposes the existence of two scales of time (an external time of the user, and an internal time of the device), then it is possible to implement accelerated computing devices which can have more computational power than Turing machines. This approach has been used to construct accelerated P systems where acceleration is obtained by either decreasing the size of the reactors or by speeding-up the communication channels [6].

In [10, 17] the time of occurrence of certain events is used to compute numbers. If specific events (such as the use of certain rules, the entering/exit of certain objects into/from the system) can be freely chosen, then it is easy to obtain computational completeness results. However, if the length (number of steps) are considered as result of the computation, non-universal systems can be obtained. Time is considered in [17, 20] as the result of the computation by using special "observable" configurations taken in regular sets (with the time elapsed between such configurations considered as output).

The authors of the current paper have also considered time to "control" the computation in two other formalisms: mobile ambients [2–4] and distributed π-calculus [13, 14]. Timers define timeouts for

various resources, making them available only for a determined period of time. The passage of time is given by a discrete global time progress function.

Acknowledgments

This work was partially supported by CNCSIS research grants IDEI 402/2007 and CNMP PARTENERIATE D1/1052/2007.

Bibliography

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. *Molecular Biology of the Cell - Fifth Edition*. Garland Science, Taylor & Francis Group, 2008.

[2] B. Aman, G.Ciobanu. Timers and Proximities for Mobile Ambients. *Lecture Notes in Computer Science*, vol.4649, 33–43, 2007.

[3] B. Aman, G.Ciobanu. Mobile Ambients with Timers and Types. *Lecture Notes in Computer Science*, vol.4711, 50–63, 2007.

[4] B. Aman, G.Ciobanu. Timed Mobile Ambients for Network Protocols. *Lecture Notes in Computer Science*, vol.5048, 234–250, 2008.

[5] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, L. Tesei. Timed P Automata. *Electronic Notes in Theoretical Computer Science*, vol.227, 21–36, 2009.

[6] C.S. Calude, Gh. Păun. Bio-Steps Beyond Turing. *Biosystems*, vol.77, 175–194, 2004.

[7] M. Cavaliere, V. Deufemia. Further Results on Time-Free P Systems. *International Journal of Foundations of Computer Science*, vol.17, 69–89, 2006.

[8] M. Cavaliere, D. Sburlan. Time-Independent P Systems. *Lecture Notes in Computer Science*, vol.3365, 239–258, 2005.

[9] M. Cavaliere, D. Sburlan. Time and Synchronization in Membrane Systems. *Fundamenta Informaticae*, vol.64, 65–77, 2005.

[10] M. Cavaliere, R. Freund, A.Leitsch, Gh. Păun. Event-Related Outputs of Computations in P Systems. *Journal of Automata, Languages and Combinatorics*, vol.11, 263–278, 2006.

[11] M. Cavaliere, C. Zandron. Time-Driven Computations in P Systems. Proceedings of Fourth Brainstorming Week on Membrane Computing, 133–143, 2006.

[12] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (Eds.). *Applications of Membrane Computing*, Springer, Natural Computing Series, 2006.

[13] G. Ciobanu, C. Prisacariu. Timers for Distributed Systems. *Electronic Notes in Theoretical Computer Science*, vol.164(3), 81–99, 2006.

[14] G. Ciobanu, C. Prisacariu. Coordination by Timers for Channel-Based Anonymous Communications. *Electronic Notes in Theoretical Computer Science*, vol.175(2), 3–17, 2007.

[15] R. Freund, M. Ionescu, M. Oswald. Extended spiking neural P systems with decaying spikes and/or total spiking. *International Journal of Foundations of Computer Science*, vol.19, 1223–1234, 2008.

[16] J. E. Hopcroft. An nlogn Algorithm for Minimizing the States in a Finite Automaton. *The Theory of Machines and Computations*, Academic Press, 189–196, 1971.

[17] O.H. Ibarra, A. Păun. Computing Time in Computing with Cells. *Lecture Notes in Computer Science*, vol.3892, 112–128, 2006.

[18] H. Lodish, A. Berk, P. Matsudaira, C. Kaiser, M. Krieger, M. Scott, L. Zipursky, J. Darnell. *Molecular Cell Biology - Sixth Edition*. Freeman, 2008.

[19] D. Molteni, C. Ferretti, G. Mauri. Frequency Membrane Systems. *Computing and Informatics*, vol.27(3), 467–479, 2008.

[20] H. Nagda, A. Păun, A. Rodríguez-Patón. P Systems with Symport/Antiport and Time. *Lecture Notes in Computer Science*, vol.4361, 463–476, 2006.

[21] Gh. Păun. *Membrane Computing. An Introduction*. Springer, 2002.

[22] Web page of the P systems: http://ppage.psystems.eu.