CS691 Report : Membrane Computing

Avadhut Sardeshmukh
Roll No 06329905
Under the guidance of Prof. Krishna S.
Computer Science and Engineering

December 5, 2007

Abstract

The main aim of this project is to survey the current state of art of the P systems (a.k.a. membrane systems) area. To this aim, a couple of introductory papers ([1], [2]) and a doctoral thesis (by Matteo Cavaliere [3]) were studied. Two main directions of research have been towards creating new models inspired from biological systems and modelling biological systems using existing models. Some of the main results obtained in both directions till now are explained and also the current open problems in this area are listed.

1 Introduction

The area of membrane systems was triggered by a landmark paper by Gheoghe Paun in 1998 in the Turku Centre for Computer Science (TUCS) Report. The same paper was later circulated in the Journal of Computer and system sciences in 2000. Till this time natural computing models were *in vitro*. For example, DNA computing. This was the first time that a cell itself was the object of research–viz. membranes in a cell and compartmentalization (*in vivo* structure of the cell).

1.1 Inspiration from Biology

It is well known that the cell is the smallest thing on earth to be unanimously considered as live.The cell has a very exquisite internal structure.

It is composed of compartments created by various membranes. An outer membrane defines the cell itself from the environment. The membrane is a semi-permeable barrier between the compartments. Molecules can be transported from one compartment to other via vesicles enclosed by membranes. The membrane structure is hierarchical. The compartments are like "protected reactors" where specific biochemical processes take place.
Our membrane computing model is roughly inspired by these properties of the cell and its internal membrane structure. The main component of this model is the membrane structure. Each membrane of this structure defines a region. Each region has a multiset of objects. And each region has a set of rules (evolution rules or symport/antiport rules) which operate on the multiset of objects in that region. Evolution rules are rewriting rules (like grammar rules) that represent the biochemical reactions going on in the cell-compartments and symport/antiport rules are transport rules that represent the vesicles through which molecules are transported from one compartment to other. The result of a computation can be the number of objects of a particular kind in a region (called output region), though there are other possibilities, too.

1.2 Formal Definition

We here define two basic models of membrane systems–those with symbol-objects and those with symport/antiport rules. Although there exists a large panoply of models of membrane systems, we believe that they can be derived/understood from these basic models.

P System with symbol objects:
[Definition] A P system of degree m ≥ 1 with symbol-objects is a tuple :

$$\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0)$$

where

- O is an alphabet and its elements are called objects
- μ is a membrane structure consisting of m membranes arranged in an hierarchical structure; the membranes (and hence the regions they delimit) are labelled with 1,2, ... ,m of μ.

- R_i are finite sets of evolution rules over O; R_i is associated with region i. An evolution rule is of the form $u \rightarrow v$, u is a string over O and v is a string over $\{a_{here}, a_{out} \mid a \epsilon O\} \cup \{a_{in_j} \mid \ a \epsilon O, 1 \leq j \leq m\}$.

- $i_0 \epsilon \{0, 1, 2, \ldots, m\}$ is the output region. In case $i_0 \geq 1$, it is the region enclosed by membrane i_0 and if $i_0 = 0$, it is the environment.

Evolution

A membrane system evolves in the following way : It starts with the multisets of objects specified by the strings $w_1, \ldots, w_m$ in the corresponding regions. There is a global clock that ticks at every time step. At each step, in each region, a multiset of objects and a multiset of rules is chosen and assignment of objects to rules is made. This is done in a *nondeterministic maximally parallel* manner. That is to say, no more rule can be added to the multiset of rules, because of the lack of objects and if there is a conflict of two rules for same (copy of an) object, then they are chosen non-deterministically. For example, if $w_i = a^5b^6c$ and the rules in R_i are $aab \rightarrow a_{here}b_{out}c_{in_2}$ and $aa \rightarrow a_{out}b_{here}$, then 2 copies of a can be assigned to one copy of first rule. Next two copies could be either assigned to another copy of first rule or to the second rule. And this choice is non-deterministic. Maximally parallel means that once four a's are consumed, no other rule can be applied to remaining objects. Thus non-deteministically, the result would be either a^3b^6c or a^2b^6c. It is important to remember that a^5b^6c represents the multiset $\{a, a, a, a, a, b, b, b, b, b, b, c\}$.

The output of a successful computation is the number of objects present in the output region in a configration such that no further evolution is possible. (i.e. no rule applicable in any membrane). Such a configuration is called halting configuration. A vector of the number of objects of each kind (e.g, no. of a's, no. of b's, etc) can also be taken as the output.

P System with symport/antiport rules:

A P system with symport/antiport rules is similar to one with symbol-objects, except that the rules are symport/antiport rules, which represent the transport of molecules through the vesicles (in the biological cell). Here, the symport/antiport rules in the set R_i are associated to the membrane i, instead of the region i. Additionally, here we have a subset E of the set of objects, whose members are supposed to exist in the environment in arbitrarily large number of copies.

The symport rules are of the form (x, in) or (x, out) and antiport rules are of the form $(x, in; y, out)$ where x, y are strings representing multisets of objects from the set O.For a symport/antiport rule, we say that the weight of the rule is $max(|x|, |y|)$ (i.e. the maximum number of objects that can be transported by the rule. Application of rule (x, in) on the membrane i means to move the objects represented by x from the region surrounding region i, into the region i. (i.e. "in" as regards the membrane i). The same is true of antiport rules also.

The output of a successful computation is defined as the number of objects present in the output region in the halting configuration. i.e. no rule should be applicable to any object in any membrane, as before.

1.3 Illustrative Example

We here give the example of a membrane system to generate the Parikh image of the language $\{a^{2^n} \mid n \geq 0\}$. This system actually combines both of the two kinds defined above. Its actually called an evolution-communication P system (section 3.1). Consider the system :

$$\Pi = (O, \mu, w_1, w_2, R_1, R_2, R_1^{'}, R_2^{'}, i_0)$$

where
$\mu = [_1[_2]_2]_1$,
$O = \{a, A, \#\}$,
$w_1 = \#$ and $w_2 = A$,
$R_1 = \phi$ and $R_2 = \{A \rightarrow AA, A \rightarrow a, \# \rightarrow \#\}$,
$R_1^{'} = \phi$ and $R_2^{'} = \{(aA, out; \#, in)\}$ and $i_0 = 2$
As can be observed, we can either double the number of A's or replace them all with a's; for, if we apply $A \rightarrow AA$ on some A's and , $A \rightarrow a$ on some, then we end up with both a and A present in region 2. But then the antiport rule $(aA, out; \#, in)$ there immediately brings in the trap symbol # and the rule $\# \rightarrow \#$ hangs the system forever. So the only successful computations will be those that produce a string of the form a^{2^n} in region 2. When we take the number of objects in region 2 as the output, we naturally get the Parikh image of this language.

2 Modelling Power of Membrane systems

As already mentioned, there have been two main directions for research in this area : viz. *i*) Propose and investigate models inspired from biological systems–from biology to mathematics and *ii*) Use the mathematical models to model/simulate biological systems and infer knowledge about their evolution–from mathematics to biology. We try to explain here what has been happening in both of these directions. In this section, we present a software simulator for membrane systems–Cyto-Sim, using which biological systems can be modelled as P-systems and their evolution observed. We give some such examples also.

2.1 The software

There are quiet a few softwares today for simulating P-systems. But we are particularly interested in showing the modelling power of P-systems, as applied to biological systems. Cyto-Sim is best to prove the point. It has been designed to implement the evolution-communication model, enriched with some probabilistic parameters inspired by cell biology.
The software simulates the standard evolution communication model of P systems. It takes as input the whole definition of a P system. Moreover, for each rule r of the system, we must specify the probability to be available. This probability is used by the software to construct a list of rules which are applicable (available) at each step.

Once this list is constructed, the assignment of objects to rules must be made, and conflicts must be solved if they arise during this assignment. As we have seen, this is done non-deterministically in our model. But while modelling biological systems, we are often aware of certain parameters which help us make this choice. These parameters can be, for example, more affinity of a certain molecule to other, or effect of concentration of enzymes on the rate of reaction, etc. So using this knowledge, we specify one more quantity for each rule r – C_{win_r} (How exactly we calculate this, is not clear at this time, though). Now, to solve the conflicts the following is done. Lets say rules $r_1, \ldots, r_k$ are competing for some symbol-object. For each r_i, the probability to win, $P_{win_{r_i}}$ is calculated as :

$$P_{win_{r_i}} = C_{win_{r_i}} / \Sigma_{j=1}^{k} C_{win_{r_j}}$$

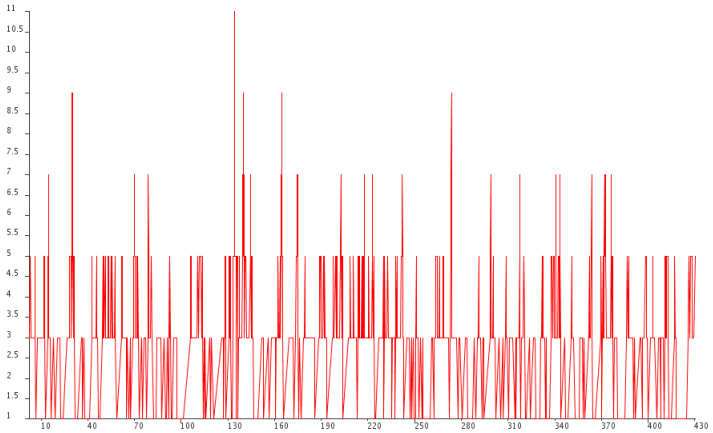
When the conflicts are resolved in this way, application of the chosen rules is

done in parallel in each region. This is done at each step during the evolution of the system.

2.2 Examples

Example 1 : Life is unpredictable

We here model the system which consists only of the rules $a \rightarrow aa$ and $aa \rightarrow a$, with a as the initial multiset in its only region (only one membrane). These rules can be looked upon as the ability of molecules to associate and dis-associate. The probability of availability is 1 for both the rules (always available) and $C_{win_{r_i}}$ for $i = 1, 2$ is 100. That is, probability to win for both the rules is 0.5 always. It may seem that this simple system reflecting life and growth is quiet stable. But as it turns out from the simulation output shown below, the system is actually unstable.



By unstable we mean the very high spikes and variations in the number of a's in the system at any given time. Thus, we have used P systems to model and conclude about a simple system that represents life.

Other semantics : An illustrative example

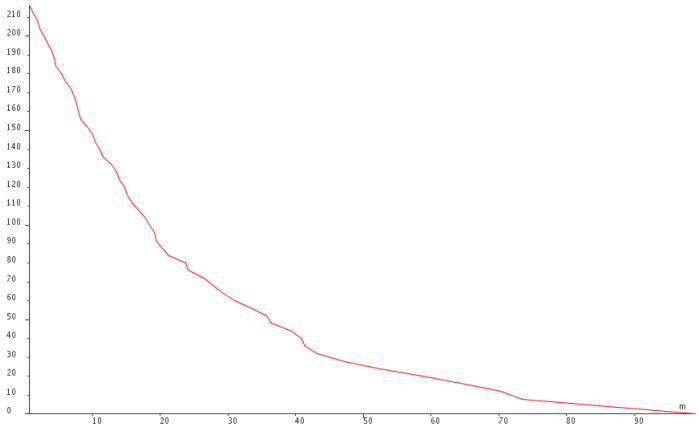
As already discussed, Cyto-Sim uses a probabilistic approach. The probabilities more often depend upon the rates and other kinetics of the chemical reactions. But there are other softwares which allow other semantics also, such as the standard maximal parallel approach. In fact we here demonstrate the difference in the output caused by the difference in the semantics of the model. The evolution of a bacteria respiratory system using the standard

maximal parallel approach can be found in [3]. And we here model the same system in Cyto-Sim (stochastic approach) and contrast the plot of evolution of the system.

The respiration in Escherichia coli is represented by a P system with only one membrane, two objects–one enzyme E (represents cytochrome bd) and the other being oxygen B. The P system has only one membrane and the only rule is $EB^{42} \rightarrow E$ which represents consumption of Oxygen (B) by the enzyme (E). That it also consumes Hydrogen and produces water is not our concern here. We are only interested in oxygen consumption (i.e. respiration). In short the system looks like follows :

$$\Pi = (O, \mu, w_1, R_1)$$

where :
$O = \{E, B\}$
$\mu = [_1]_1$
$w_1 = (E)^k B^j$
$R_1 = \{EB^4 2 \rightarrow E\}$
The Cyto-Sim code for this system is straight-forward. We do not recall it here. We directly present the output, as before.



We particularly want to highlight that this curve (using Cyto-Sim) is an exponential curve whereas the plot corresponding to the standard maximal parallel approach is a straight line with slope very close to our curve. So one possible conclusion could be, due to the kinetic rates taken into consideration, the Cyto-Sim model is more realistic whereas the maximal parallelism has abstracted out these details, making the curve a straight line.

2.3 Conclusion

P systems can be used to model various real biological systems and the software can be enhanced to produce more meaningful results which can help biologists. We also observe that investigating the difference in output due to the difference in semantics of the models is an open problem. To this aim, Cyto-Sim can be further modified/enhanced to allow other semantics such as the maximal parallelism.

3 Computational Power of Membrane systems

3.1 Evolution communication P systems

3.1.1 Introduction

The evolution communication model is a composite of the two basic models defined in section 1.2 : P system with symbol-objects and P system with symport/antiport rules. But here the evolution rules on symbol-objects do not have target indications such as $\{here, out, in\}$, because the transportation is done by symport/antiport rules here. We have already seen an example of such a system in section 1.3.

Another variant : Symport/Antiport of rules

In biological cell, due to movement of molecules, the reactions taking place in a compartment change over time. Symport/antiport of rules is inspired by this biological reality. We actually define a P system in which the rules can be moved by symport/antiport and objects evolve only through evolution rules. The formal definition follows :

[Definition] A P system with symport/antiport of rules or CR P system (Communication of Rules) is defined as a tuple

$$\Pi = (O, R, l, \mu, w_1, \ldots w_n, R_1, \ldots, R_m, R_1^{'}, \ldots, R_m^{'}, i_0)$$

where

- O is the set of objects
- R is a set of simple evolution rules (i.e. no target indications)
- l is an injective labelling of rules in R; let L be the set of all labels

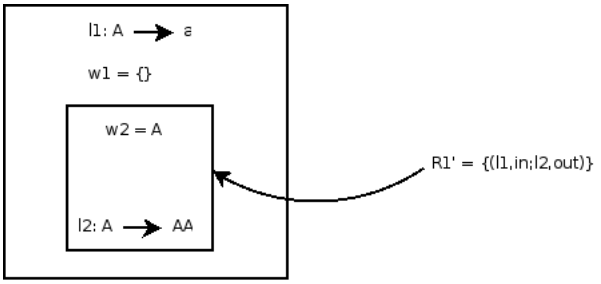
- w_i, $1 \leq i \leq m$ is the initial multiset of objects present in region i
- $R_i \subseteq R$, $1 \leq i \leq m$ is the set of simple evolution rules associated to each region i
- R_i', $1 \leq i \leq m$, is the set of symport/antiport rules associated to region i. The strings in symport/antiport rules (i.e. x or y in (x, in) or $(x, in; y, out)$) are over the set L. That is they represent a list of rules to be moved.
- i_0 is the output region

Note that here the symport/antiport is applied on the rules themselves, so x in (x,out) is a string of labels of rules, representing *set* of rules, not a multiset, because rules cannot exist in multiple copies, like the objects.

Now, an evolution step consists of a mixed application of evolution and symport/antiport rules. There may be conflicts such as a rule $u \to v$ (with label, say, l_j) is applicable to a multiset of objects u in its current region, as well as it is a candidate for transport as in, a rule such as (l_j, out). These two rules have the same "priority", meaning this choice is again done in *non-deterministic maximally parallel* way.

Illustrative Example

We present a P system with symport/antiport of rules to generate again the language $\{a^{2^n} | n \geq 0\}$. Consider the system pictured below :



Observe that in region 2 the rules $A \to AA$ and $(l_1, in; l_2, out)$ conflict and there is a non-deterministic choice to either continue doubling the number of A's or stop and convert them into a's (by using the rule $A \to a$ which comes in by the antiport rule above). Clearly, this generates the Parikh image of the language $a^{2^q n}$. Note the difference between the membrane system for the same language given in section 1.3 and this one.

3.1.2 Universality Results

The Evolution-Communication P systems become universal with 3 membranes allowed. The corresponding result is :
$PsECP_3(1, 1, ncoo) = PsRE$
The meaning is, EC P systems with maximum of 3 membranes, symport and antiport rules of maximum weight 1 and only noncooperative rules (i.e. the left side is of length one) can generate the family of Parikh images of RE languages. Here,the output of the system is defined as the vector of multiplicities of objects of each kind from the set O present in the output region in a halting configuration.

This is proved by constructing a P system of given size to simulate a programmed grammar. That class of programmed grammars characterises the RE languages is known already.

P systems with symport/antiport of rules:
As we have seen with an example, a P system with symport/antiport "of" rules was able to generate a language such as a^{2^n}, with only antiport rule of weight one and two membranes. Indeed, they become universal with three membranes and antiports of length 2. The result is :
$PsCRP_3(0, 2, cat_1) = PsRE$
This means CR P systems with at most 3 membranes, no symport rules, antiport rules of weight at most 2 and using only 1 catalyst generate the family of Parikh images of RE languages.

This has been proved by simulating a matrix grammar with appearance checking (in Z-binary normal form) with a membrane system of given size.

3.2 Time-free P systems

3.2.1 Introduction

Till now we have been assuming that their is a global clock that ticks at each time unit and time required to complete one step is same for all the rules. All of them start at the start of the unit, and finish when the clock ticks, and then the next step starts. This is not the case usually, in biological systems. In reality, different biochemical reactions take different time to complete depending upon factors that are often unpredictable.

So we need a model of the membrane system which assumes nothing about the completion times of different rules (i.e. associates with each rule a completion time to each rule) and produces the same result in spite of

changes in this assignment. That is, whatever may be the assignment, the system generates the same language, always. We call such systems as Time free P systems.

As can be seen, the main issue here is achieving synchronisation. There are various ways to do this. The important ones are using signalling rules bi-stable catalysts, and using only communication (i.e. symport/antiport). Universality results have been proved for these approaches. The formal definitions follow :
[Definition] A P system with signal-promoters and bi-stable catalysts, is a construct

$$\Pi = (O, C, D, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, R_1^s, \ldots, R_m^s, i_0)$$

where

- C is the set of bi-stable catalysts
- D is the set of signal-promoters
- The rules in R_i are of two types : (i) non co-operative, that is of the form $a \rightarrow v$ and (ii) co-coperative using bi-stable catalysts, that is of the form $ca \rightarrow cv$, $ca \rightarrow \bar{c}v$, $\bar{c}a \rightarrow cv$ or $\bar{c}a \rightarrow \bar{c}v$, where a belongs to $O - C \cup D$ and v is a string over a_here, a_out, a_in_j for all such a's
- R_i^s are sets of signalling rules which are of the form $a \rightarrow v|_z$ or $ca \rightarrow cv|_z$ where a and v are as before and z is a string over $\{p_{here}, p_{out}, p_{in_j}\}$ for all signal-promoters, p
- All other components remain the same

Now, a timed P system is a system Π defined as above with a time for completion assigned to each rule r by the mapping

$$e : R_1 \cup \ldots \cup R_m \cup R_1^s \ldots \cup R_m^s \rightarrow N(Natural\ numbers)$$

We assume that there is a global clock that ticks at each step marking units of equal length. If execution of rule r starts at step j, then it ends at step $j+e(r)$. During this time, the objects being used by this rule are unavailable to other rules in the region.

At each step, all active rules take part in the assignment process. The assignment, as before, is done in a non-deterministic maximally parallel way.

All evolution rules are always active. But a signalling rule $u \rightarrow v|_z$ is active only when the signalling objects specified by z are present in the region. After the evolution of u into v, the objects z are moved according to their target indications. Note that the signalling objects do not exist in multiple copies and they can only be moved, neither created, nor destroyed.

All said, now we can say what exactly a time-free P system is : Its a P system such that for any mapping e as discussed above, it generates the same language (same set of vectors of numbers). We would now like to characterise which P systems are time free. An observation is as follows:
Every P system using only non-cooperative evolution rules is time-free

This is clear if we note that the evolution using only non-cooperative rules can be visualised by a tree whose nodes are objects and yield is the output. Basically there are no conflicts arising out of timing issues–all evolutions are sort of sequential.

Illustrative Example
We again take the same language as an example – $\{a^{2^n} | n \geq 0\}$. We define a time-free P system to generate this language. Consider the system Π defined as below:

$$\Pi = (O, C, D, \mu, w_1, w_2, R_1, R_2, R_1^s, R_2^s, i_0)$$

where : $O = \{a, b, p\}$,
$C = \phi$, $D = \{p\}$,
$\mu = [_1[_2]_2]_1$, $w_1 = bp$ and $w_2 = a$,
$R_1 = R_2 = \phi$,
$R_1^s = \{b \rightarrow b|_{(p,in)}, b \rightarrow b|_{(p,out)}\}$ and $R_2^s = \{a \rightarrow aa|_{(p,out)}\}$,
$i_0 = 2$
It is very easy to see that this indeed generates the desired language. The rule which doubles the number of a's in region 2 is activated by the signal promoter p. At each step, in region 1 there is a non-deterministic choice to send p in or out. In case it is sent in, the process of doubling the number of a's is iterated. In the other case, p is sent to the environment, ending the computation.

A more important observation is that this is a time-free system. Because, the rules are applied in mutual exclusion (there is only one signal promoter that activates **all** the rules), no matter what time they take to complete, the language generated remains the same. (One more reason is, the only rules used are non-cooperative).

3.2.2 Universality Results

The universality results for time-free P systems, obtained thus far are :

- $fPsP_1(2cat_*, 0) = PsRE$
  This means that time-free systems using unbounded number of bistable catalysts with at most 1 membrane and no signal promoters can generate the class of Parikh images of RE languages. This is shown by simulating a matrix grammar with appearance checking in Z-binary normal form with a time-free P system using bi-stable catalysts, of the said size. This result is obtained for catalysts which are not bistable, as follows.

- $fPsP_2(cat_*, *) = PsRE$
  This is easy to understand. For, we can simulate the behaviour of a bistable catalyst using a simple (mono-stable) catalyst, if we have enough signal promoters, which we do have in this case. For simulating the behaviour of a bi-stable catalyst say, c_j, we need two signal promoters p_j and $\overline{p_j}$ which activate and de-activate the new rule (which has only monostable catalyst) according as the state of the bistable catalyst (i.e. c_j or $\overline{c_j}$).

- $fPsPP_1(1, 2) = PsRE$
  This is a new class of time-free systems. Namely, the second approach to achieve the synchronisation – through communication (symport/antiport). It is same to the symport/antiport P system except the fact that now it has the time parameter taken into consideration. That is, a symport/antiport rule r has a time $e(r)$ associated to it. and objects being moved by r are not available (cannot be used by other rules) till this much time after the rule r starts executing, and also that a subset E_{inf} of the set of symbol objects is believed to be available in the environment in arbitrary number of copies.

  They are proved to be universal by simulating the register machine using a time-free P system. To do this, we represent the value contained in each register by the multiplicity of a particular kind of object. (i.e. for each register, we have a symbol-object type). Now there are only these operations of addition and subtraction by 1 and halting. The

addition operation is simulated by bringing in one object of the required kind (the one that represents the register being implemented) and subtraction by throwing out one object.

3.3 P/O systems : The Evolution/observation model

3.3.1 Introduction

The Evolution-Observation model is kind of a conclusion to all the material presented till now. We saw that we can "learn" from biology and we can give to biology. The P/O model does exactly that. The evolution and observation model is a pair of two less powerful systems–one is a biological system which just "lives" its own life and other is a computing device which observes this evolution of "life" and interprets the behaviour of the biological system to produce results. Thus, we can on one hand, observe the system and decide if something "unpleasant" happens in the system, through the observer. And, on the other hand, we look at the whole thing as a computing device, where, depending upon the states through which the biological system goes, the observer produces certain strings.

We can apply this approach to any biological system, by modelling it (using say, membrane systems, or splicing systems etc), giving an input to the model and observe the evolution using a finite automaton. When this approach is applied to membrane systems, the model created is called as a P/O system. The definition follows :
[Definition] A P/O system is a pair $\Omega = [\Pi, A]$ where Π is a P system and A is an observing multiset finite automaton, with output alphabet Σ. The language generated by the system is

$$L(\Omega) = \{A(s) | s \ \epsilon \ B(\Pi)\}$$

Where, the behaviour $B(\Pi)$ of the of the P system is the set of all possible *sequences* of configurations during any computation (evolution) of the system Π. In short, the language contains all the words the observer produces when run on the sequence of configurations of the P system.

Now, the multiset finite automata(MFA) works on a multiset (represented by a string) of objects and to each final state, a symbol from the output alphabet Σ is associated as its label. A successful computation produces the label of the final state it halts in as the output. All other computations produce λ. We already know that a configuration of a P system is a vector

of strings representing multisets of objects in each region of the system. For example if at some point in a P system (with two membranes), the content of region 1 is two a's and three b's and the content of region 2 is one a and two b's, then the configuration is represented by the vector (a^2b^3, ab^2). But here, for the sake of simplicity we think of it as only one string (or only one multiset). By subscripting a symbol object with the region it is in, we remove the need to consider different strings for each region. So, in the above example, we will represent our configuration as $a_1^2a_2b_1^3b_2^2$. So the output of the MFA when it is run on the string representing a configuration c as above, is referred to as $A(c)$. And the output generated when it is run on a sequence of configurations $c_1, \ldots c_m$ is the concatenation of the strings produced when the MFA is run on c_1, c_2, etc, separately. That is, the output is $A(c_1), \ldots A(c_m)$. It is important to note that the MFA is run **afresh** on each of the configurations, i.e., every time, starting from the initial state, and the output is a pure concatenation of individual, independent outputs.

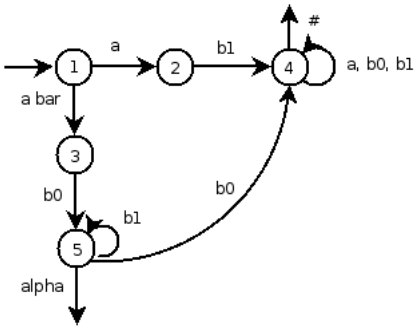
Illustrative Example

We finally present a P/O system to generate the Parikh image of our favourite language $\{a^{2^n} | n \geq 0\}$. Note that this language is non-context-free while the P/O system that we use has as its components, a simple P system with symbol-objects and non-cooperative rules and a deterministic MFA, both of which are have power less than context-free.

Consider the P/O system $\Omega = (\Pi, A)$ where,

$$\Pi = (O, \mu, w_0, w_1, R_0, R_1)$$

Such that, $O = \{a, b\}$, $\mu = [_0[_1]_1]_0$, $w_0 = \lambda,\ w_1 = a$,
$R_0 = \{b \to \lambda\}$,
$R_1 = \{a \to aa, a \to b, b \to b, b \to b_{out}\}$
And the MFA A defined over the input alphabet $V = \{a, b_0, b_1\}$ (because that all where the objects a and b can belong) is as shown below :



The output alphabet of the MFA is $\{\alpha\}$. When run on the halting computations (i.e. sequences of configurations, starting from the initial configuration and leading to a halting configuration) of the P system, the finite automata actually generates the language $\{\alpha^{2^n}\}$.

Note that the states for which outputs are defined are the final states. And there is a symbol # that is produced in a "undesired" final state and it is not a part of the output alphabet so the configuration which generate that symbol, are not included in the final language. Now, the P system starts with a in its region 1 and the only applicable rules are $a \to aa$ and $a \to b$, which conflict. But if the second rule is chosen, then the system ends up in a configuration where there is a b in region 1 and this configuration produces the trap symbol # on the MFA because of the path towards right–the even-states-path(remember that the output of a sequence of configurations is the individual configurations' outputs concatenated), and thus is discarded. So the only successful computations do $a \to aa$ for some n number of times. After this, there are 2^n a's in region 1 and there is a choice to stop. To do this, all of the a's have to be converted to b's, all at once. (Otherwise again, a configuration like $a^j b$ will lead to a trap). When region 1 contains only b's, the odd-states-path of the MFA takes over the control(because there are no a's in the system). The b's are taken one by one to region 1, by applying $b \to b_{out}$ on one b and $b \to b$ on all others. This is ensured by the MFA as follows : look at the transitions, a configuration where there are no a's (tested by saying $\overline{a}$ leads to the state 3. Now, the case where there are more than one b's in region 0 (ie. configurations of the form $b_0^j b_1^k$) will again lead us to the trap state 4 (so what through the good accepting state 5!). Only if there is only 1 b in region 0 and all others in region 1 is the state 5 reached and it produces one α per consumed b. The b in region 0 is immediately deleted. A sequence of configurations such that every time one b is moved from region 0 to region 1, thus produces 2^n α's on the MFA's outputs. Note how the observer MFA controls which computations of the P systems are to be taken into considerations and which of them dumped.

Universality of P/O systems :

It has been proved that P/O systems with very simple components are capable of being universal. For example the kinds of P/O systems used in the above example, where the P system is a simple system with symbol-objects and non-cooperative rules and the observer is a deterministic MFA, both less powerful than context-free, have been proved to be universal. There are plenty of other results, but we do not call them here.

4 Conclusion

The two main conclusions are :

- We can learn from biology
  The time-free systems and the P/O systems are good examples of the fact that we can develop computationally complete, powerful theoretical models by looking at the biological systems. And not only the *in vitro* structure but also the *in vivo*. The evolution/observation approach can also be applied to splicing systems to define yet more powerful models. Many of the results achieved till date remain open for improvement. The number of membranes, weights of symport/antiport rules, catalysts, etc. can add to the power of a membrane system to a large extent.

- We can help biology
  The Cyto-Sim software shows a pathway towards this. We can not only model the biological systems using mathematical models, but also infer knowledge from them using mathematical techniques to observe their evolution. The Cyto-Sim software can now be enhanced to draw more specific conclusions about biological systems, or make some predictions about their real evolution. One important conclusion was that during this modelling, maximal parallel approach might not help always. It might abstract out some essential real details. For this, the kinetic rates (stochastic) model should be used while modelling.

References

[1] Gheorghe Paun. Introduction to membrane computing. 1998.

[2] Gheorghe Paun and Carlos Martin-Vide. Elements of formal language theory for membrane computing. *Reseach group on Matematical Linguistics, Report GRLMC 21/01.*

[3] Matteo Cavaliere. Evolution, communication, observation:from biology to membrane computing and back. *Ph.D thesis, University of Seville*, November 2005.

[4] Arto Salomaa. *Formal Languages*. Academic Press, NY and London, University of Turku, Turku, Finland, first edition, 1973.

[5] Matteo Cavaliere and Sean Sedwards. Modelling cellular processes using membrane systems with peripheral and integral proteins. *Technical Report 07/2006, The Microsoft Research-University of Trento Centre for Computational and Systems Biology*, July 2006.