ANALYSIS OF P-SYSTEMS UNDER A MULTIAGENT SYSTEMS PERSPECTIVE

Alberto Arteta, Angel Goñi, Juan Castellanos

Abstract: *Membrane computing is a recent area that belongs to natural computing. This field works on computational models based on nature's behavior to process the information. Recently, numerous models have been developed and implemented with this purpose. P-systems are the structures which have been defined, developed and implemented to simulate the behavior and the evolution of membrane systems which we find in nature. What we show in this paper is an application capable to simulate the P-systems based on a multiagent systems (MAS) technology. The main goal we want to achieve is to take advantage of the inner qualities of the multiagent systems. This way we can analyse the proper functioning of any given p-system. When we observe a P-system from a different perspective, we can be assured that it is a particular case of the multiagent systems. This opens a new possibility, in the future, to always evaluate the P-systems in terms of the multiagent systems technology.*

Keywords: *P-systems mapping, multiagent systems.*

Conference: *The paper is selected from Seventh International Conference on Information Research and Applications – i.Tech 2009, Varna, Bulgaria, June-July 2009*

Introduction

Natural computing ia a new field within computer science which develops new computational models. These computational models can be divided into three major areas:.

- Neuronal networks.
- Genetic Algorithms
- Biomolecular computation.

Membrane computing is included in biomolecular computation. Within the field of membrane computing a new logical computational device appears: The P-system. These P-systems are able to simulate the behavior of the membranes on living cells. This behavior refers to the way membranes process information. (Absorbing nutrients, chemical reactions, dissolving, etc)

Membrane computing formally represents, through the use of P-systems, the processes that take place inside of the living cells. In terms of software systems, it is the process within a complex and distributed software. In parallel computational models, p-systems might be as important as the Turing machine is in sequential computational models.

In this paper, we design a multiagent system that it is capable to simulate the behavior of any given P-system. After this we will study the advantages of applying this system. By simulating the P-systems with a multiagent system, we can analyze the P-system from a different perspective. Therefore establishing a procedure that will focus on the correction of any given P-system based on the multiagent system paradigm.

In order to do this we will take the following steps:

- Introduction to P-systems theory;
- Components of a particular multiagent system;
- Link between P-systems and multiagent systems;
- Example and code;
- Conclusions and further work.

Introduction to P-systems theory

In this section we will study into detail all of the theories related to the paradigm of the P-systems. A P-system is a computational model inspired by the way the living cells interact with each other through their membranes. The elements of the membranes are called objects. A region within a membrane can contain objects or other membranes. A p-system has an external membrane (also called skin membrane) and it also contains a hierarchical relation defined by the composition of the membranes. A multiset of objects is defined within a region (enclosed by a membrane). These multisets of objects show the number of objects existing within a region. Any object 'x' will be associated to a multiplicity which tells the number of times that 'x' is repeated in a region.

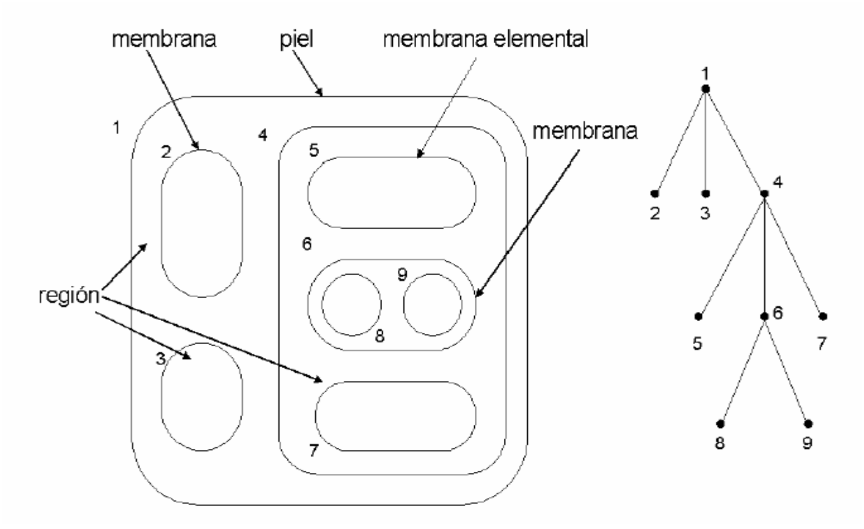


Fig. 1. The membrane's structure (left) represented in tree shape (right)

According to Păun 's definition, a transition P System of degree n, n > 1 is a construct: [Păun 1998]

$$\Pi = \left(V, \mu, \omega_1, .., \omega_n, (R_{1,}\rho_1), ..(R_{n,}\rho_n), i_0\right)$$

where:

1. *V* is an alphabet; its elements are called objects;
2. µ is a membrane structure of degree n, with the membranes and the regions labeled in a one-to-one manner with elements in a given set ; in this section we always use the labels 1,2,..,n;
3. $\omega_i \; 1 \le i \le n$, are strings from V^* representing multisets over V associated with the regions 1,2,..,n of µ
4. $R_i \; 1 \le i \le n$, are finite set of evolution rules over V associated with the regions 1,2,..,n of µ; ρ_i is a partial order over $R_i \; 1 \le i \le n$, specifying a priority relation among rules of R_i . An evolution rule is a pair (u,v) which we will usually write in the form $u \to v$ where u is a string over V and v=v' or v=v'δ where v' is a string over $(V \times \{here, out\}) \cup (V \times \{in_j \; 1 \le j \le n\})$, and δ is a special symbol not in. The length of u is called the radius of the rule $u \to v$
5. i_o is a number between 1 and n which specifies the output membrane of Π

Let *U* be a finite and not an empty set of objects and N the set of natural numbers. A *multiset of objects* is defined as a mapping:

$$M : V \to \mathrm{N}$$

$$a_i \to u_1$$

Where a_i is an object and u_i its multiplicity.

As it is well known, there are several representations for multisets of objects.

$$M = \{(a_1,u_1),(a_2,u_2),(a_3,u_3)...\} = a_1^{u_1} \cdot a_2^{u_2} \cdot a_n^{u_n} ......$$

Evolution rule with objects in *U* and targets in *T* is defined by $r = (m, c, \delta)$

where $m \in M(V), c \in M(VxT)\ and\ \delta \in \{to\ dissolve, not\ to\ dissolve\}$

From now on *'c'* will be referred to as the consequent of the evolution rule *'r'*

The *set of evolution rules* with *objects* in *V* and targets in *T* is represented by R *(U, T).*

We represent a rule as:

$x \rightarrow y \quad or \quad x \rightarrow y\delta$ where x is a multiset of objects in M((*V*)xTar) where Tar ={here, in, out} and y is the consequent of the rule. When δ is equal to "dissolve", then the membrane will be dissolved. This means that objects from a region will be placed within the region which contains the dissolved region. Also, the set of evolution rules included on the dissolved region will disappear.

P-systems evolve, which makes it change upon time; therefore it is a dynamic system. Every time that there is a change on the p-system we will say that the P-system is in a new transition. The step from one transition to another one will be referred to as an evolutionary step, and the set of all evolutionary steps will be named computation. Processes within the p-system will be acting in a massively parallel and non-deterministic manner. (Similar to the way the living cells process and combine information).

We will say that the computation has been successful if:

- The halt status is reached.
- No more evolution rules can be applied.
- Skin membrane still exists after the computation finishes.

Multiagent System Technology

The multi agent system we are going to study is a system made of several entities of computers. These systems can achieve goals unreachable for single agent systems (one agent system). The main characteristcs of multiagent systems are:

- Proactivity;
- Autonomy.

Formally an agent is a real or virtual entity that:

1. Is able to act within a given environment.
2. Is able to communicate with other agents.
3. It has its own resources.
4. Is able to retrieve information and to (at least partially) know the environment.
5. It can reproduce.

According to the definition of agent, a multiagent system can be defined as a system of computers which contains the following elements:

1. An environment *E*, which is a space.
2. A set of objects $O \in E$.
3. A set of agentes $A \in O$.
4. A set of relations *R* between objects and agents.
5. A set of operations that allows to the agents interact with the objects.

When studying the P-system under a multiagent system technology we will consider *A=O.* All of our agents will relate with each other and will communicate with each other.

In the next sections we will see that p-systems can be studied under muliagent systems technology by performing some transformations. The goal of this paper is to provide more properties to the p-system than the ones they have nowadays.

Link from P-systems to Multiagent Systems

Once we have seen the structures of p-systems and multiagent systems separately, we just need to establish a link between these two universes. Once we build this link from one universe to the other, we will be able to see p-systems under a different perspective and therefore treat them differently.

Due to the fact that a p-system and a multiagent system have several components, it makes sense trying to establish a relation between the two of them. Thus, the first step is changing the perspective of the membranes. The way to do this is:

In a p-system, given a set of membranes $M = \{m_i \mid i \in \mathrm{N}, 1 \le i \le n\}$ where m_i is a membrane, we establish a a function f_{agent} which is defined as:

$$f_{agent} : M \to A$$
$$f_{agent}(m_i) = a_i \ \forall i \in \mathrm{N}\ i \le n, \quad \text{n number of membranes}$$

Under this new perspective, every agent $a_i \in A$ has a set of resources which includes:

- Multiset of objects within the region enclosed by the membrane $m_i \in M$
- Set of evolution rules within the region enclosed by the membrane of $m_i \in M$

Besides, each membrane $m_i \in M$ which contains another membrane $m_j \in M$ can be mapped as father-son relationship within a multiagent system.

In order to set up the synchronism in our system, we need to have an agent called $a_{sync} \in A$. This agent will make sure that the membranes are synchronized.

The p-system will consider three major stages:

1. Static structure of the p-system
2. Dynamic behavior of the p-system
3. Synchronism between membranes.

Now is when we can describe the mapping:

Given a set of membranes $M = \{m_i \mid i \in \mathrm{N}, 1 \le i \le n\}$ and $W = \{\omega_i \mid i \in \mathrm{N}, 1 \le i \le n\}$ the set of multisets of a p-system where $\omega_i \in W$ is the multiset of objects representing atoms, molecules included in the membrane $m_i \in M$ Let R_i be a finite and non empty set of evolution rules belonging to the membrane $m_i \in M$ and let u be the structure of membranes that define a father-son relation between membranes*.*

We can see a membrane $m_i \in M$ as $m_i = (\omega_i, R_i, children)$ where ω_i is the multiset of objects and R_i the set of evolution rules. *Children* are the membranes which are contained in $m_i \in M$ and also *children* are a subset of the set of membranes *M.*

The mapping between p-systems and multiagent systems can be built from a function and a proper operator:

$$f : M \times \{\omega_1, \omega_2, .., \omega_n\} \times \{R_1, R_2, .., R_n\} \to MAS$$

Where MAS is the multiagent system associated to the p-system. This multiagent system contains the agents a_i, resources R_i and a set of operations Op_i

This function f can be defined in terms of 3 other sub-functions. These three sub-functions are:

1. $f_{name}: \mathrm{N} \rightarrow Agentname$
   $f_{name}(i) = a_i$
2. $f_{resource}: \{\omega_1, \omega_2, .., \omega_n\} \times \{R_1, R_2, .., R_n\} \rightarrow \mathrm{Resources}$
3. $f_{operation}: dynamic\ transition \rightarrow Agent\ behavior$

Every agent a_i will be associated to a set of resources called $\mathrm{Res_i}$ and set of operations called a_i. When referring to a_i it will mean a_i with its resources and operations.

Formally speaking, the multiagent system associated to a p-system with n membranes will have the set of resources as the union of all the objects, and the union of all the set of evolution rules included in every membrane i.e $\mathrm{Resources} = \left[\bigcup_{i=1}^{n} R_i \right] \cup V$ where R_i is the set of evolution rules included in the membrane i.

By using this information we can provide more details of $f_{resource}$

$f_{resource}$ will convert a multiset of objects ω in a string like this:

The objects are represented as (x_i, n_i), where x_i is the object and n_i its multiplicity.

Thus, $f_{resource}$ will convert a multiset of objects:

$\omega = (x_1, n_1), (x_2, n_2) ...., (x_n, n_m) \Rightarrow f_{resource}(\omega) = x_1 x_1 .. x_1 x_2 x_2 .. x_2 ..$ Where every object x_i appears n_i times.

And given an evolution rule:

$$r = (x_1, n_1), (x_2, n_2) ...., (x_n, n_m) \rightarrow (x_1, m_1), (x_2, m_2) ...., (x_n, m_m)$$
$$\Rightarrow f_{resource}(r) = \left(x_1 .. x_1 x_2 .. x_2 .. x_n .. x_n,\ x_1 .. x_1 x_2 .. x_2 .. x_n .. x_n \right)$$

where n_i is the multiplicity of the object x_i in the antecedent and the multiplicity m_i in the consequent. $f_{resource}(r)$, has a two dimensional image. The first component of the image is a conversion of the antecedent into a string where every object x_i appears n_i times. The second component of the image is a conversion of the consequent into a string where every object y x_i appears m_i times, $n_i, m_i \in \mathrm{N}$.

Once the mapping of the static parts has been developed, we are going to establish the necessary operations for the multiagent system to evolve in the same way that its associated p-system does. In order to achieve this we create an operator that returns the status of the transition of a p-system to a specific time. This synchronization must take place, and therefore it will have to be a part of the set of resources of the multiagent system. In order to study this synchronization, we create a new resource called *sync*. This resource is:

1. An integer (computing step)
2. A letter (Status)

The initial transition status is the integer 0.

$sync_i$ = {transition status of the agent a_i}

The possible status in every computing step could be:

A. Rules election,

B. Objects consumption,.

C. Communication between membranes

We will also create an agent called a_{sync} which will be in charge of assuring the synchronization along with the agents (associated to the membranes).

For instance, let us have a set of transition statuses:

$$sync_1 = 3A \quad sync_2 = 2B, \quad sync_3 = 4C$$

Our agent a_{sync} will make sure this situation never happens. It will assure the following:

$$sync_i = sync_j \; \forall i \neq j \; i, j \in \mathrm{N}$$

At last, the set of the union of all possible transition statuses will be part of the set of resources of the multiagent system.

Initially $sync_i = 0 \quad \forall i \leq n \quad i \in \mathrm{N}$.

So far, we have seen that components of p-systems are mapped to multi agent system components (static part of a given p-system). Moreover, the evolution of a transition status with membrane systems can be also mapped as an evolution process within the multiagent systems. Because of this, we can state that modelling membrane systems can be seen as a multiagent system. This multiagent system has objects, agents, resources and operations.

Let us have three membranes m_1, m_2, m_3 and m_1 contains m_2 which contains m_3. In our multiagent system this situation we would have three agents a_1, a_2, a_3 where a_1 is in the father-son relationship with a_2 which is in the father-son relationship with a_3. It is represented as follows:

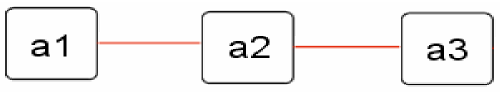


Fig. 2. Three membranes m_1 contains m_2 which contains m_3 in the MAS.

Another way of representing the transformation of a set of membranes into MAS is a three of agents. In the three the root nodes are the fathers. The level below indicates the sons which can also be fathers of several agents. Given a generic set of membranes the mapping into MAS is as follows:

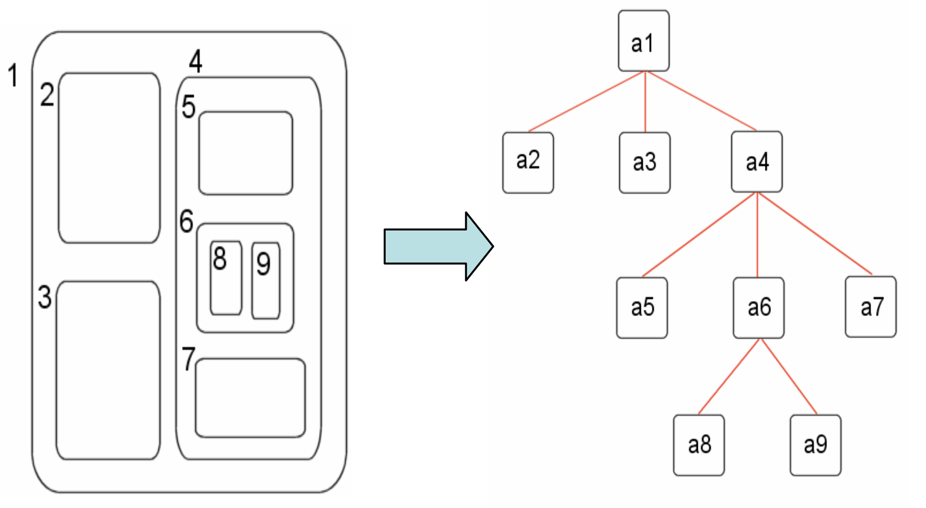


Fig. 3. Representation of a set of membranes into MAS.

in the way that a p-system evolves, it would be performed similarly in the MAS. In every evolution step the agent a_{sync} makes sure that the transition status of the membranes are the same. This assures the system evolves properly. In our MAS we will also have the following possible status:

1. Evolution rules selection
2. Evolution rules application
3. Communication between membranes

Example and code:

Once is clear the definitions and meanings of the P-systems and MAS and it is also clear the way we build the link between them we are going to show the transformation of a p-system into a MAS. So whenever an action occurs in a given p-system there will be a parallel action occurring in the MAS.

Let us suppose we have a p-system containing three membranes. This p-system is able to calculate the square of a given number. We are going to see 2 evolution steps (the initial one and the final one) in the p-systems can be transformed into MAS. To understand p-systems in detail, refer to [Arroyo, 2001].

a) Static components mapping

The P-system has:

- A set of membranes M=$\{m_1, m_2, m_3\}$
- An Alfabet V={a,b,c,d,e,f}
- A set of multiset of objects $M(V)=\{\omega_1=\{\},\omega_2=\{\},\omega_3=\{af\}\}$ where ω_i is the multiset of objects within the region delimited by the membrane m_i $\forall i \in \mathrm{N}\, i \leq 3$.
- A Multiset of evolution rules

$$R(U,T)=\left\{\begin{matrix} R_1(U,T)=\{\mathrm{e}\rightarrow \mathrm{e_{out}}\}, R_2(U,T)=\{\mathrm{b}\rightarrow \mathrm{d}, \mathrm{d}\rightarrow \mathrm{de}, (\mathrm{ff}\rightarrow \mathrm{f} > \mathrm{f}\rightarrow \delta)\}, \\ R_3(U,T)=\{\mathrm{a}\rightarrow \mathrm{ab}, \mathrm{a}\rightarrow \mathrm{b}\delta, \mathrm{f}\rightarrow \mathrm{ff}\} \end{matrix}\right\}$$

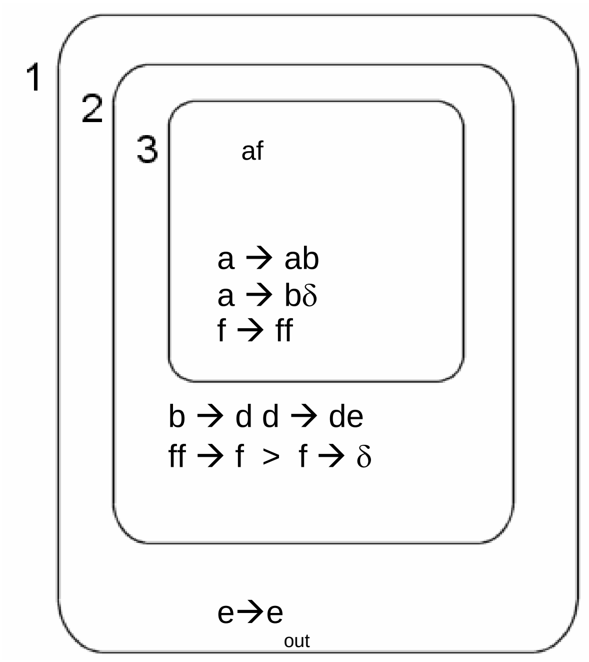


Fig 3 P-system which calculates a random square number.

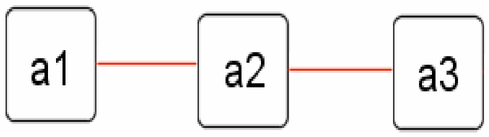


Fig 4. Associated Multisystem Agent.

The set of agents A= $\{a_1, a_2, a_3, a_{sync}\}$

The resources used by the agent a_i are referred as Res_i, these are the multiset of objects and set of evolution rules.

$$\left\{\bigcup_{i=1}^{n}\mathrm{Res}_i\right\}=\{\{[af],[(a,ab),(a,b\delta),(f,ff)\},\{[],[(b,d),(d,de),(ff,f),(f,\delta)]\},\{[],[(e,e_{out})]\}\}$$

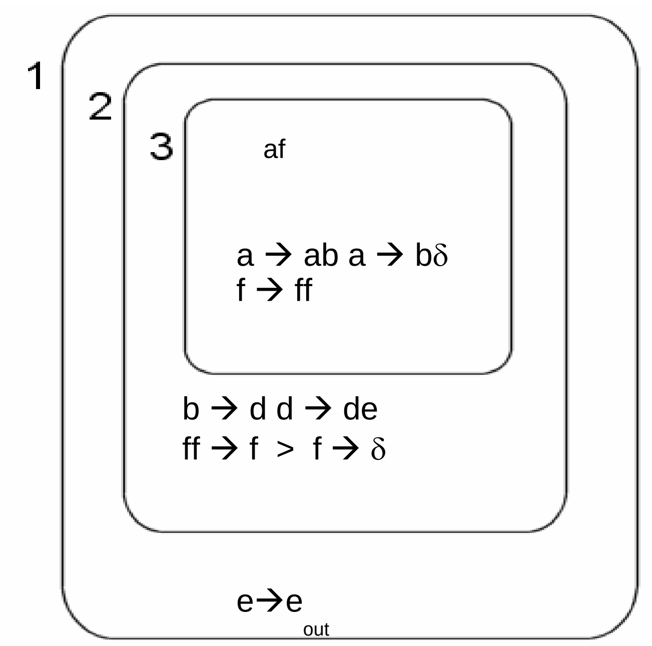
b) Dynamic behavior

In order to see how MAS evolves, we need to focus on:

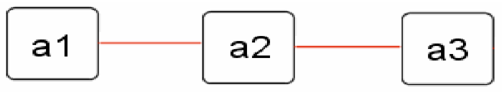
- Rules election
- Objects consumption
- Communication Stage

Initial Transition Status $sync_i = 0 \quad \forall i \le 3 \quad i \in \mathrm{N}$ This condition is checked by the agent a_{sync}

In the transition status 1), the p-system evolves, In the Region 3 the rule number 1 and number three are applied.



Moreover, in our MAS:



,

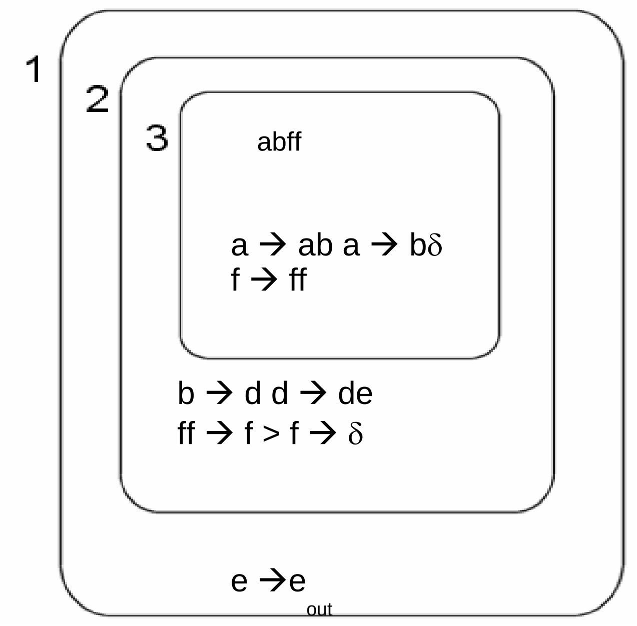
our agent a_{sync} makes sure that $sync_i = 1A \quad \forall i \le 3 \quad i \in \mathrm{N}$. In every region we are analysing the rules to be applied. Every agent selects from its resources the rules to be applied.

When this is done, a_{sync} assures we go to the next status. $sync_i = 1B \quad \forall i \le 3 \quad i \in \mathrm{N}$. Here the agents execute the action “apply rules”.

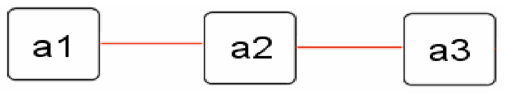
Following our example, only agent a_3 selects the rules r_1 *and* r_3 from its set of resources $\mathrm{Re}\ s_3$.

After applying rules a_{sync} assures all the agents are synchronized and go to the next status. $sync_i = 1C \quad \forall i \leq 3 \quad i \in \mathrm{N}$. Now is when the agents to communicate with each other and exchange objects if needed.

After the first computing step, we have:



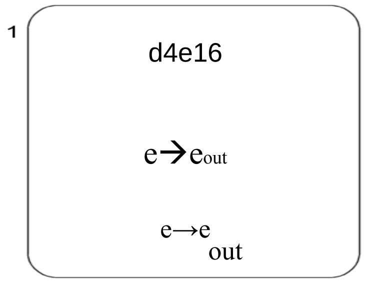
in our p-system, and our MAS is



where the resources used by the agents would be:

$$\left\{\bigcup_{i=1}^{n}\mathrm{Res}_i\right\}=\{\{[abff],[(a,ab),(a,b\delta),(f,ff)\},\{[],[(\mathrm{b,d}),(\mathrm{d,de}),(\mathrm{ff,f}),(\mathrm{f},\delta)]\},\{[],[(\mathrm{e,e_{out}})]\}\}$$

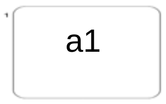
By repeating this process will reach a situation like the following:



Note. The number next to the object indicates the multiplicity of the object.

which will return $16=4^2$

In our MAS, we will end up with the following situation:



The rest of the agents have been eliminated when the action “dissolve” was performed.

$$\left\{\bigcup_{i=1}^{n}\text{Res}_i\right\}=\{\{[d4e16],[(\text{e},\text{e}_{\text{out}})]\}\}$$

Code:

A possible code to implement the MAS associated to the p-system is:

```
Rules R
Objects w
Dissolve = false
While NOT Dissolve
    waitsync
    //EVOLVE
    R' = Rules_election(w,R)
    w' = Rules_Application (w,R,P)

        waitsync
    //COMUNNICATION
    w = COMMUNICATION(w')
    dissolve = finish(w)
```

waitsync: Sinchronization between agents

Rules_election: Selection of the rules

Rules application: Application of the rules

communication (w) Communication between agents and exchange of the objects w

Finish: The computation is finished

Conclusion

In this paper, we have studied some topics of membrane computing. As a part of this study, we have explained some concepts of the p-systems. Concepts such as:

- Components
- Interactions between the components.
- The evolution of a p-system.

Moreover, we have focused our work on some concepts of the multiagent systems such as objects, agents, relations and resources. We have then established a link that connects both paradigms. Because of this we have proved that p-systems are a particular case of a multiagent system. With links such as the one we have built, we are able to to describe any p-system in Multiagent system terms. Thus, the study of the p-systems can also be included in the study of the multiagent systems.

It is not intended to explain deeply neither the P-system technology nor the Multiagent system technology. There are many papers that have been published about that [Păun 1998], [Arroyo,2001] [Arroyo, 2003], The P-system described here has been generic. We have not explained into detail aspects as rules election, priorities between rules, etc. The general and main concepts have been useful to show our work. Again, MAS has been generally described without focusing in every detail. In order to understand better the concepts of Multiagent system technology, refer to [Wooldridge,2002] ”An Introduction to Multiagent Systems”, Wiley, 2002. or [J.Ferber, 1999] “Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence”. The Multiagent system described here has been generic.

We have built a “bridge” between these two universes and, because of that we have added some extra properties to the p-systems. The code provided here is a proposal but it is not the main goal of this work. By formally establishing an association between the P-systems and MAS we take a step on opening the p-systems to other technologies.

Proposals to create a link between P-systems and MAS exist already, Our proposal has been inspired in [Acampora, 2007]. However, our model has not been focused on the implementation but on providing new aspects to the current P-systems. In order to do this we have provided more details in the design of our mapping.

According to our association between p-systems and multiagent systems, a cell membrane can be treated as an independent and autonomous agent. By providing autonomy and independence to the cell membranes, we also change the perspective for the rest of the components within the p-systems. For example, when a membrane *'A'* contains another membrane '*B'*, a dependency from '*B'* to *'A'* exists. This occurs because, whenever 'B' is dissolved, its objects become objects of the membrane 'A'. By looking at this scenario from a multi agent perspective, these two membranes are two agents that establish a relation. Because of this relation, the agent 'B' provides objects to the agent 'A'. This happens whenever the agent 'B' executes an action called 'dissolve.'

This perspective allows individual treatment of the p-systems components.

Nowadays we work with the p-system as an entire compacted block of components that are going through an evolutionary process. The p-system functioning is treated under a global perspective.

Under the MAS perspective, the analysis changes. The p-system works properly if and only if every agent $a_1, a_2, .., a_n, .., a_{sync}$ works properly.

Thus, the analysis of the correct functioning of a given p-system is not based on the global processes but in the correction of all the agents which are part of the MAS. When every agent is assured of functioning correctly, we are assured that the entire multiagent system is working correctly.

By studying the characteristics of the p-system under the multi agent system perspective we can add the p-systems as a particular case of the current MAS.

This new perspective can also change the implementation of the p-systems. The use of the agent a_{sync} assures the synchronization of the entire multi agent system. This characteristic does not exist in the current implementation of p-systems.

Because the Multi agent systems are used broadly, they can be applied to many current systems. It is possible to state that, by using mappings such as this, we can reduce more systems into MAS. The key for this will be updating the rules in the MAS. When this occurs we would be able to reduce more systems into MAS and to assure correct functioning.

Bibliography

[Păun 1998] “Computing with Membranes”, Journal of Computer and System Sciences, 61(2000), and Turku Center of Computer Science-TUCS Report nº 208, 1998.

[Acampora, 2007]A Proposal of Multi-Agent Simulation System for Membrane Computing Devices Giovanni Acampora, Member, IEEE and Vincenzo Loia Member, IEEE 2007 IEEE Congress on Evolutionary Computation (CEC 2007).

[Jimenez, 2006]Membranes as Multi-agent Systems: an Application to Dialogue Modelling Gemma Bel-Enguix and Dolores Jimenez Lopez Bel-Enguix. G., Lopez, D.,T., 2006, in IFIP International Federation for Infonnation Processing, Volume 218 Professional.

[A. Arteta, 2008] “Algorithm for Application of Evolution Rules based on linear diofantic equations” Synasc 2008, Timisoara Romania September 2008[1] A. Syropoulos, E.G. Mamatas, P.C. Allilones, K.T. Sotiriades “A

[Arroyo, 2001] “Structures and Bio-language to Simulate Transition P Systems on Digital Computers,” Multiset Processing (. [Arroyo, 2003] “A Software Simulation of Transition P Systems in Haskell, Membrane Computing,”

[Wooldridge,2002] ”An Introduction to MultiAgent Systems”, Wiley, 2002.

[Acampora, 2005] “Fuzzy control interoperability and scalability for adaptive domotic framework,” in IEEE Transactions on Industrial Informatics, vol.1, no. 2, May 2005, pp. 97–111.

[J.Ferber, 1999] “Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence,” Addison Wesley, 1999.

Authors' Information

Alberto Arteta Albert – Associate professor U.P.M Crtra Valencia km 7, Madrid-28031, Spain;
e-mail: aarteta@eui.upm.es

Research: Membrane computing, Education on Applied Mathematics and Informatics

Angel Goñi – Researcher natural computing group U.P.M, Madrid, Spain; e-mail: anandgel@gmail.com

Juan Castellanos – Professor (UPM) Department Artificial Intelligence. Faculty of Informatics, Madrid 28660, Spain, e-mail: jcastellanos@fi.upm.es