A Multiagents system for a parallel and distributed paradigm

Arteta, Assistant Professor, Computer Science Troy University, Alabama

Juan Castellanos, C. Nuria Gomez, Lf Mingo. Associate Professors Polytechnic University of Madrid

***Abstract*—** Cell computing works on computational models based on cell's behavior to process the information in a faster way than usual. Recently, numerous biological models have been developed and implemented. In particular cell membrane systems are structures that simulate the behavior and the evolution of membrane systems found in Nature. This paper proposes an application capable to supervise a cells membrane model through a multiagents system (MAS). By using multiagent systems the membrane models functionality improves. This creates the possibility of having scenarios where biological systems and multiagent systems work together to enhance the performance in bio-inspired models.

***Keywords*—** Multiagent systems, membrane computing, supervised learning, biological models, P-systems.

I. Introduction

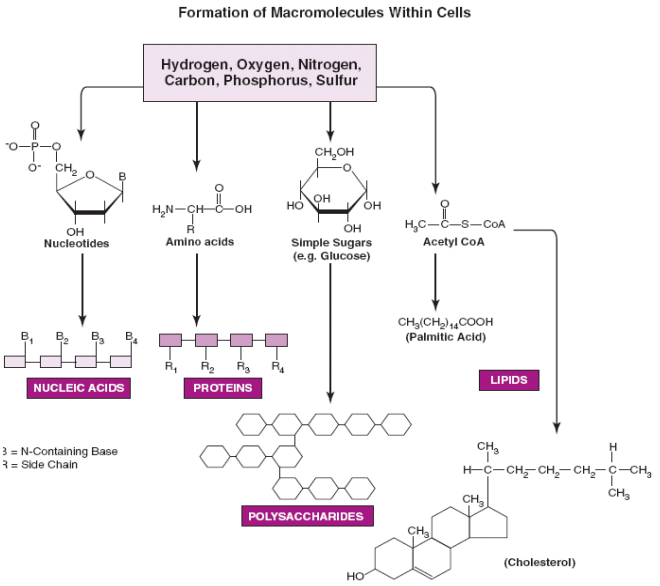
The concept of transition p-system was first introduced in a 1998 by Gheorghe Paun, whose last name is the origin of the letter P in 'P Systems'. Variations on the P system model created what we call 'membrane computing.'

This model was inspired by biology, however P systems were used as a computational model, instead of a biological model. Nowadays there are some attempts of creating new biological models through the use of p-systems.

In [1] there are some keywords related to membrane computing such as parallelism, distributed applications, algorithms. Paun avoids stating that there are implementations of p-systems. This happens because the model he proposes has an inherent parallelism and non determinism. Those characteristics cannot be implemented in a conventional computer. [1].

Regarding simulators, [2] remarks: simulations on conventional computers can be useful too: In fact all these programs were used on biological applications and they can also play an important role for didactic purposes and to establish new researching areas.

The living cells we see that they process food and nutrients in a very characteristic way. Living cells get atoms or molecules and then react by producing other atoms and molecules.



This process has been studied and it has also been tried to implement into computers. Living cells act and process nutrients according to certain rules that take place in a parallel and in a non deterministic manner. This model has been copied by many researchers to establish new machines that process information in the same way. In 1998 George Paun studied the described biological model and then was able to create what we call nowadays 'Membrane computing'. This was created mainly inspired on the processes that take place inside of the living cells. After analyzing such processes, it is noticeable that they have properties that can be used into a computational model. This model was a revolution because it opened new researching areas for solving NP-complete problems.

Foundations and organizations were aware of this new revolutionary way of computing and encouraged the research on that direction. Examples of these are EMCC (European Molecular Computing Consortium) and The Consortium for Biomolecular Computing.

In 2004 the official p-system web http://psystems.disco.unimib.it was created; since then, it has been supported by the EMCC

In February 2003 membrane computing was defined as "the most revolutionary way of computing within ICT". This was said by the Institute of Scientific Information.

The transition P-systems have been suffered a transformation themselves. This transformation comes from the desire of collecting more biological properties. Moreover the use of automats has been added to improve the performance of p-systems. One of the main goals for the p-systems to achieve is the resolution of Np completes problems in polynomial time. Natural computing group in Madrid is currently working on implementing p-systems in specialized hardware. The parallelism degree achieved is high and the temporary results obtained are promising. Also, the idea of reducing to practice the p-systems is more realistic.

Algorithms for applying evolution rules are getting improved too. A new algorithm that obtains optimal results when simulating the rules application phase, is about to be published. This algorithm use combinatorial laws and statistics.

Np complete problems have been solved in polynomial time and sometimes even in linear. We are working closer with biological areas and now there is a membrane computing application in biology. It is believed that membrane computing can be a reliable framework to model living cells processing.

Moreover, this group has avoided approaching p-systems as an isolated technology. has worked on scenarios in where the p-systems can work together with multiagent systems. There is another paper to be published in where p-systems works with autonomous robots to find optimal solutions to known problems.

II. NATURAL COMPUTING

Natural computing is the science which develops new computational models based on the way Nature works. These computational models are classified in three fields:

- Biomolecular computation. [1]
- Genetic Algorithms [3]
- Neuronal networks[4].

Membrane cells computing are included in bimolecular computation. A new logical computational device appears: The P-system. These P-systems are able to simulate the behavior of the membranes on living cells when absorbing nutrients, producing chemical reactions, dissolving membranes, etc)

Membrane computing formally represents, through the use of P-systems, the processes that take place inside of the living cells.

This works shows a multiagent system that it is able to conduct the functionality of a given P-system to resolve complex problems in fast way.

Contents of this paper are classified as follows:

- Introduction to cell membranes theory.
- Components of a particular multiagent system
- Proposal for a multiagent system which controls a transaction P-system
- Particular Case : Robots
- Example and code
- Conclusions and further work

III. CELL MEMBRANES THEORY

This section introduces the paradigm of the P-systems (Membrane computing). 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 / membrane can either contain objects or other membranes.

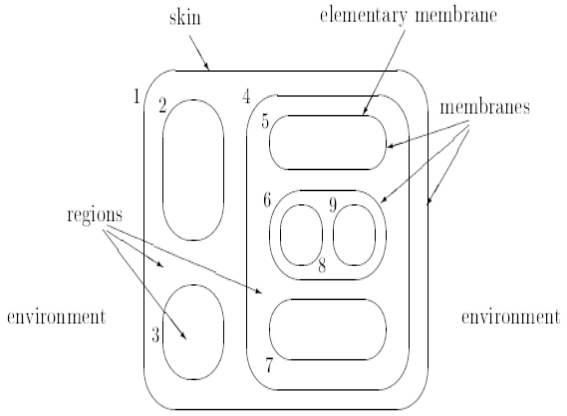


Fig. 1. The membrane's structure (left) represented in tree shape (right)

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

$$\prod = (V, \mu, \omega_1, .., \omega_n, (R_1, \rho_1), .. (R_n, \rho_n), i_0)$$

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:

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

$$a_i \rightarrow u_1$$

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

$$m \in M(V), c \in M(V \quad) a \quad \delta x \in \{n \; d \; T \quad d, n \quad t \; sd \; o \quad s\}$$

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).*

Rules are represented as:

$x \rightarrow y \quad o \quad r\, x \rightarrow y\delta$ 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 making its set of evolution rules 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 a new transition is generated. The step from one transition to another one is defined as an evolutionary step, and the set of all evolutionary steps is 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).

The whole info is processed successfully if:

The halt status is reached.
No more evolution rules can be applied.

III IMPLEMENTATIONS

This section has the goal to show techniques that implement the functionality. We are showing here two kinds of algorithms: Parallel and sequential algorithms.

The parallel algorithm is suitable to implement it in ad hoc hardware. In the other hand the sequential one works when using either universal hardware or ad hoc hardware with universal components.

Both algorithms work to implement the idea of pure non determinism. Since there are no non-deterministic computers [Hwang, 1994], the idea of implement non deterministic algorithms running on those computers is impossible. However it is possible to obtain some degree of non determinism. [Arteta, 2009]

The algorithms presented here are shown in pseudo-code . It is intended to stress the idea of the universality of the algorithms. They must work regardless the code they are built on.

Efficient algorithms can be translated to multiprocessor machines. From there they can work on real parallel Computers.

- **Sequential algorithms:**

Currently, a numerous sequential algorithms have been implemented. As an example the sequential one called step by step is described in this chapter.

1) Sequential algorithm step by step Paso a Paso (sPP)

In literature about simulation tools of membrane systems , it is widely used an algorithm in which rules are applied one by one in a set of micro-steps of evolution, in particular, during the process of calculating the multiset of evolution rules. This algorithm implements the random election of one of the active evolution rules, the modification of the multiset of objects presents in the region, and finally, the new calculation of the set of active evolution rules in the region. This algorithm is expressed below, where ω_S is the multiset of objects existing in a region, R_A is the set of active evolution rules, and ω_{R_A} is the resulting multiset of evolution rules to be applied in order to make evolve the region.

The algorithm is finite because any evolution rule has a maximal applicability benchmark over the multiset of objects [Fernandez,2006]. Hence, application of any evolution rule over a multiset of objects drives to the non-applicability condition eventually. At this point, the rule will be removed from the set of active evolution rules]. This fact implies that the set of evolution rules will eventually be the empty set; hence the loop will finish. The algorithm shows certain degree of non determinism

Clearly the algorithm is non deterministic and the election of the active evolution rule to be applied is random.

Her we present a trace example for the algorithm’s execution:

Input*:*

$$\omega_S = a^{22} \cdot b^4 \cdot c^{21} \cdot d^{22}$$

$$input(R_1) = a^2 \cdot b^0 \cdot c^4 \cdot d^2$$

$$input(R_2) = c^0 \cdot b^0 \cdot c^2 \cdot d^1$$

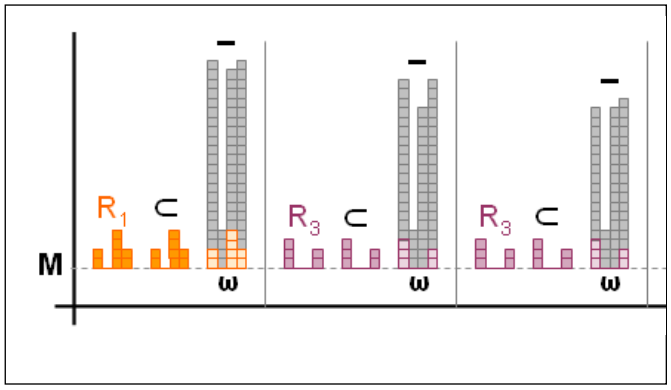
$$input(R_3) = a^3 \cdot b^0 \cdot c^0 \cdot d^2$$

$$input(R_4) = a^1 \cdot b^1 \cdot c^2 \cdot d^1$$

Output*:*

$$\omega_S = a^2 \cdot b^0 \cdot c^1 \cdot d^4$$

$$\omega_{R_A} = R_1^{\;2} \cdot R_2^{\;2} \cdot R_3^{\;4} \cdot R_4^{\;4}$$



Step by step.

Fig2. Execution trace of the algorithm called 'Step by Step'

As shown, the process is iterative and it involves two or three actions. In every iteration an evolution rule is randomly selected; then the algorithm checks that the rule is useful and applicable [1]. If so, objects from the multiset of objects are removed.(as many as the number of objects included in the antecedent of the selected evolution rule) .This way, the number of objects of the multiset will reduced in the next iteration. The algorithm finishes when there are no more applicable evolution rules.

```
{1} REPEAT
{2}   i ← random(1,|R_A|)
{3}   IF NOT imput(R_A.reglas[i]) ⊂ ω_S
THEN
{4}     R_A ← R_A − {R_A.reglas[i]}
{5}   ELSE BEGIN
{6}     ω_S ← ω_S − input(R_A.reglas[i])
{7}     ω_{R_A} ← ω_{R_A} + {(R_A.reglas[i],1)}
{8}   END
```

Step by step Sequential Algorithm's code

IV. MULTIAGENT SYSTEM TECHNOLOGY

Multi agent system [5] [6] can reach goals that are impossible for single agent systems (one agent system).
The main properties of multiagent systems are:

Proactivity
Autonomy

Formally speaking 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. Have their own resources.
4. Is able to retrieve information and to (at least partially) know the environment.
5. Can reproduce.

According to the definition of agent, a multiagent system is defined as a system of computers which containing 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
6.

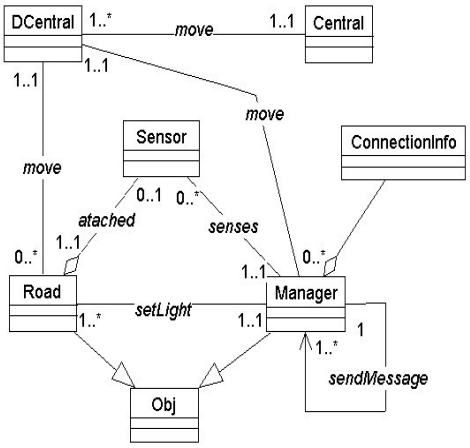


Fig 3. Example of a workflow of traffic control through Multi-agent System

7.

V. AGENTS LEADING THE MEMBRANE MODEL: PROPOSAL

Once p-systems and multiagent systems are described separately, this section shows a way to create a multiagent system that supervise and control the membranes operations during the computation process. This agent supervised system will be referred as MSSA [7],[8] (Membrane system supervised by agents) from now on. As there are different components in a membrane system it is necessary now to establish how the multiagent system can manage the whole model. The following proposal is inspired in the model in [7] .

- 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, For any membrane mi it is necessary to define a single agent, . This can be defined as an injective function. $f_{agent} : M \to A$

  $f_{agent}(m_i) = a_i \ \forall i \in \mathrm{N}\ i \le n,$

- $a_i \in A$ In this way, every single agent is in charge of a single membrane. The agent is called membrane-agent
- The Multiset of objects within the region enclosed by the membrane $m_i \in M$ and the rules to to be applied on them are supervised by the membrane-agents.All agents relate and communicate with each other.

In order to set up the synchronism in our system, a synchronization agent called $a_{sync} \in A$ is needed. This agent ensures a proper synchronization between the membrane agents. The Multiagent system has to supervise the 2 major processes occurring in the membranes model. These are:

Dynamic behavior of the p-system (Computation and communication)
Synchronism between membranes.

Let us define each agent.

$f_{n\ a} : \mathrm{N} \rightarrow A$ g n e a

$f_{n\ a}(i) = a_i$

$f_{r\ e} : \{\omega_1, \omega_2, \ \omega_n\} \times \{R_1, R_2, \ R_n\} \rightarrow \mathrm{R}$, s oe

$f_{o\ p} : d_e\ r\ y_a t\ m_i\ o\ a \rightarrow A$ n h g s e Every agent a_i is linkedto a set of resources called Res_i and set of 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

$$\mathrm{R}\ s\ eo\ u = \left[\bigcup_{i=1}^{n} R_i\right] \cup Vs$$

where R_i is the set of evolution rules which are included in the membrane i.

The multiagent system contains the agents a_i , resources R_i and a set of operations Op_i

Now let us define the multiagent system to evolve in order to control the transition P-system. This uses an operator that returns the status of the transition of a p-system to a specific time. In order to do this we create a new resource called *sync* which is defined as:

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

The initial transition status is the integer 0

The System status for every step is defined as a letter (A,B or C), meaning as follows :

A. Rules election,
B. Objects consumption,.
C. Communication between membranes

The synchronizing agent ensures:

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

$s\ y_i = s\ y_j \ \forall i \neq j, \ i, j \in \mathrm{N}$

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

Example:

Let us have three membranes m_1, m_2, m_3 and m_1 c o m_2 w t ach io in m_3. The multiagent system has 3 membrane agents a_1, a_2, a_3 where a_1 represented as follows:

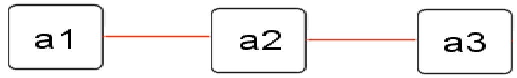


Fig.4. Three membrane agents.

Below there is a diagram describing the relationship between a membrane system (left) and the supervisor Multiagent system.

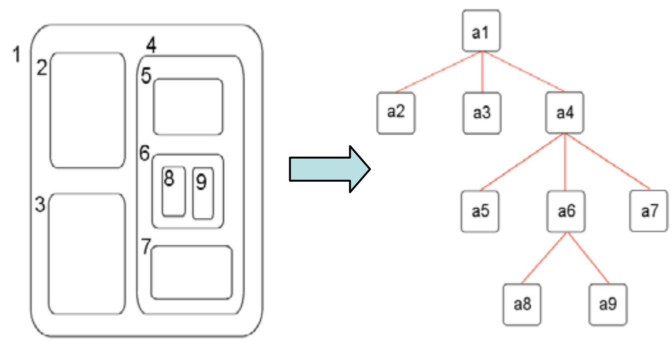


Fig. 2. MMAS description

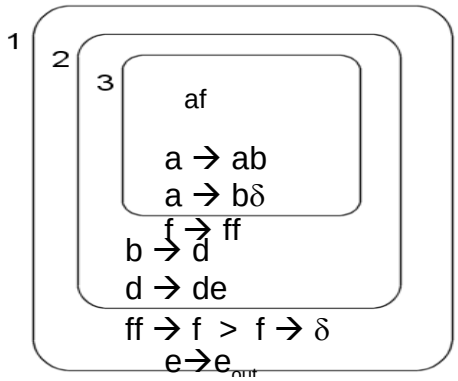
Example of a computation process supervised by a Multiagent system

For this example a 3 membrane p-system has been chosen. This p-system is able to calculate the square of a given number. [1]. There are 2 evolution steps described[9] (the initial one and the final one) The P-system acts according the Multiagent system instructions.

a) Components

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)= $\{\omega_1 = \{ \ \omega_2 = \{ \ \omega_3 = \{a \ \}\}$ 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)= $\left\{\begin{array}{l} R_1(U,T) = \{e \rightarrow e_o\ \}\ R_2(U,T) = \{b \rightarrow d\ d \rightarrow d\ (\ \rightarrow f\ f \rightarrow f\}, \\ R_3(U,T) = \{a \rightarrow a\ \ a \rightarrow b\ f \rightarrow f\ \},\ \ f \end{array}\right\}$



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.

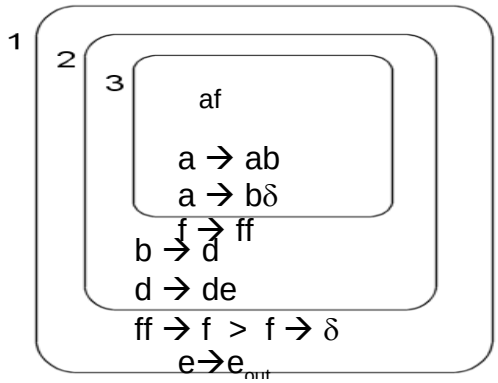
Now we go step by step

- Rules election
- Objects consumption
- Communication Stage

Initial Transition Status $s \quad y_i = 1B \; c \forall i \leq 3 \quad i \in \mathrm{N}$

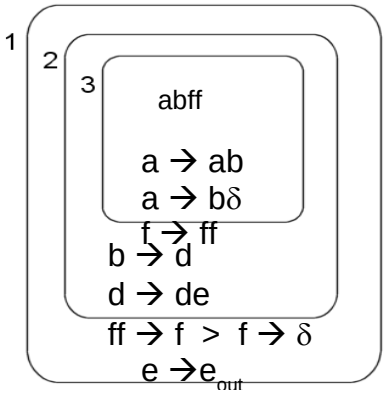
This condition is checked by the agent a_{sync}

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

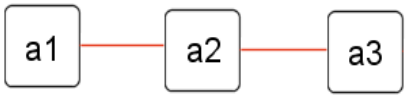


The agent a_{sync} makes sure that $s \quad y_i = 1A \; c \forall i \leq 3 \quad i \in \mathrm{N}$. In every region the candidate rules to be applied are analyzed. Then every agent selects from its resources the rules to be applied.

In the mean time a_{sync} ensures the synchronization between regions. $s \quad y_i = 1B \; c \forall i \leq 3 \quad i \in \mathrm{N}$. Here the agents execute the action “apply rules”. agent a_3 makes the P-systems choose $r_1 \; and \; r_3 \quad \mathrm{Res}_3$ .After applying rules a_{sync} assures all the agents are synchronized. i.e. $s \quad y_i = 1C \; c \forall i \leq 3 \quad i \in \mathrm{N}$. After the first computing step, the P-system looks like this:



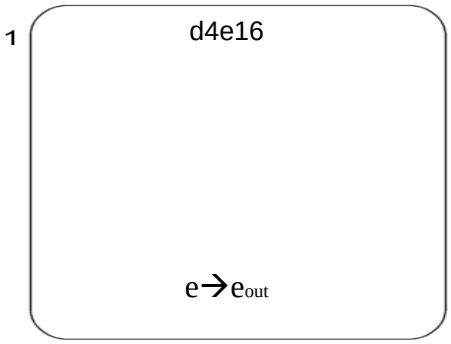
P-system controlled by agents.



The resources used by the agents are:

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

In the end the program ends like follows:



Note. The number (4) besides the object (d) indicates the multiplicity of the object

The system returns 16= 4^2 . The agents have supervised every single computing step and have ensured a minimum number of operations to optimize the functionality of the membrane system. The main difference with the standard P-

system is that elections are chosen in an intelligent way by the agents who supervise the membrane model

V. PARTICULAR CASE MULTIAGENTS: ROBOTS

This section propose the implementation of the MMAS for a particular case of Multiagents: Autonomous robots. The design of an autonomous robot that it is capable to simulate the behavior the living cells to solve known problems through the use of P-systems. Although this has been done so far by traditional P-systems [Paun, 98] a new model is proposed in this section. A design of several autonomous robots that behave as the living cells when processing information. This way these robots can obtain solutions to known problems. The interesting part here is that performance improves by reducing computational complexity. As mentioned before, traditional P-systems are the structures used within membrane computing to do the living cells simulation. P-systems evolve in a parallel a non-deterministic way. By using an autonomous robot, it is possible to eliminate the non-determinism as the evolution will be orientated to obtain results in a faster way. In order to do this the following steps would be necessary to take. In the section there are different topics:

1. Description of autonomous robot
2. Theoretical model of a robot processing information from the living cells;
3. Traditional P-system vs. P-systems with robots;

Presently, humans and robots are used to interact with each other. It could be said that robots try to simulate humans' actions. Part of those humans' actions is making decisions. If a robot can make decision, would it be correct to state that the robot is intelligent? Today, there is no a unique definition for intelligence. However it seems clear that the concept: "intelligence" [Brooks, 1986] is related to:

- Understanding
- Problem solving
- Learning.

Even when it is possible to create robots that solve problems it is not clear that they either understand or learn. That is the reason why the definition of intelligence does not cover all the possible meanings. Although it is uncertain to state that robots are intelligent, it is possible to establish a classification for them:

In general terms it is possible to separate the robots into two classes:

4. Deliberative robots
5. Reactive Robots

The first kind is the traditional one. This types of robots work based on scheduled actions and known information. Reactive robots interact with the environment, they observe, process information, and make decisions based on the environment they are.

From our perspective the second type is the interesting one. These robots can adapt to the environment they live and make decisions based on that. Not only making decisions but also solving problems. Adaptability is other characteristic of intelligence.

Another characteristic of traditional robotics is the use of a centralized system. This centralized system stores all the information of the environment. The information is represented in a symbolic way. After processing the information is possible to calculate the next action. The problem here is when the environment changes constantly. That is why traditional robots become useless in this scenario. The aim is trying to build a robot able to adapt to its environment regardless the type of environment it is. In that way, the robot will have some basic actions and it will create a bigger knowledge by learning from the environment. The learning process comes from the reactions that robots do.

Learning process is a complex one:

If a robot is created to experiment in a lab, it is possible to determine all the situations in where the robot is going to be. On the contrary, if the robot created wants to be useful in the real world, there is no way for the creators to determine how many different situations the robot is going to face. Thus, the creators must be able to design a robot that learns and therefore be able to react correctly when a new circumstance arises. These are the different ways for implementing the learning process.

Generallly speaking there are four types of learning.

- Auto-organized: It works with random variables.
- Supervised: any action has different data and variables.
- Hybrid: A combination of the two previous ones
- Feedback: The response from the environment influences on its actions.

The robots proposed in this section will learn by receiving feedback from the environment.

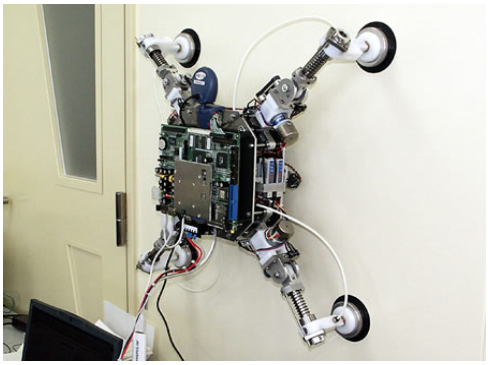


Fig 5 Spider robot

VI.

VI. PARTICULAR CASE MULTIAGENTS: ROBOTS

Once the characteristics of P-systems and autonomous robots are explained, a proposal for building a community of robots

that acts as a P-system. In particular t let us say as a P-system able to learn; or even better an "Intelligent P-system".
As there are regions in the living cells a robot will be allocated in every region delimited by a membrane.
The way to do this is:
In the regions, given a set of membranes $M = \{m_i \mid i \in \mathrm{N}, 1 \le i \le n\}$ where m_i is a membrane, a robot is allocated. Thus, a new function must be defined for each membrane or region

f_r ; $M \to A$ o t

f_r $(m_i) \to R_i \ \forall i \in \mathrm{N}\ i \le n,$ n n m fm

The P-system considers three major stages:

- Static structure of the P-system
- Dynamic behavior of the P-system
- Synchronization between membranes.

The static structure of the P-systems is updated by the existence of a robot r_i

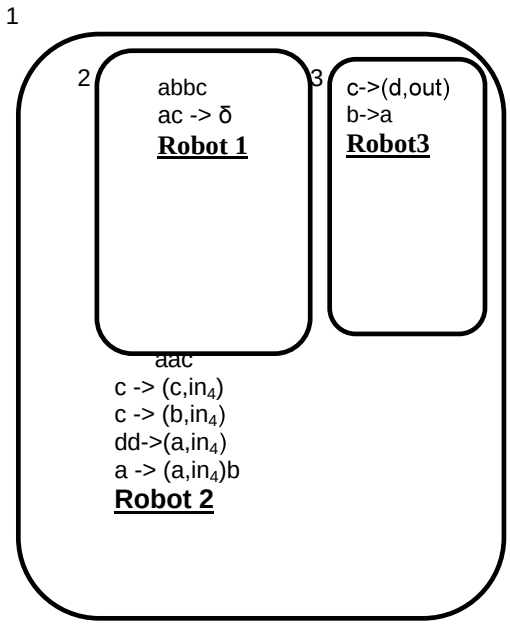


Fig 6. P-system is controlled by reactive robots.

In the figure above, the robot outside (Master Robot) controls the evolution of the P-system.

Every robot controls a region. The robot is storing the information about the information processing for every region. The P-system evolves following the robots patterns. For every region the robot store information and makes decisions based on the information is stored. Moreover the robot outside the P-system controls the execution steps of the P-system.

Robots learn from the times that evolution rules are applied and the results obtained. The aim of this P-system supervised by robots is not just to find solutions to known problems, but also improving performance on that.
According to Paun's model all the rules application processes occur in every region in a parallel manner. Moreover the process is non-deterministic.

Every robot R_i collects information about the membrane m_i in where he is allocated.

- The evolution rules that are applied in m_i and number of times that every rule r_i is applied to obtain an extinguished multiset

For every region the robot R_i stores records as: $(r_1,1)\ (r_2,2)\ ,(r_t,7.)\ ,,(r_m,3),$

- The robot outside the P-systems:

- Process the information arriving from all the robots.
- It keeps the number of computation steps until the program finishes.
- Stores information of the communication phase between membranes
- It tells the robots what decisions to make in order to reduce computation steps. (Deterministic way)

The more time the P-system works the better results are obtained. At the end the robot will tell the robot in each membrane which evolution rules should be applied and also how many times those rules must be applied in order to obtain fast results. This will reduce the execution time.

Traditional P-systems versus P-systems with robots

Implementing membrane computing with robots shows a few advantages compared to the traditional ones.
The election for the evolution rules to be applied would be taken according the robot's indications. At first, robots will not be very accurate on the elections but during the learning process programs will take shorter to finish and to find solutions. Finding solutions in a faster way requires certain degree of what it is referred as "intelligence". P-systems with robots can be referred as intelligent P-systems. Although this is the main advantage to obtain when using P-systems with robots, there are certain disadvantages too:

- The need of auxiliary space to store information about the rules' election
- The need of extra time to calculate the times that the evolution rules have to be applied
- In the beginning the results might take longer than when using traditional P-systems. This might occur because as there are not enough information from the P-system computation, the decisions made by the robot about the rules could be worse than electing rules in a non- deterministic way (as the traditional P-systems do)

- When changing non-determinism by intelligent elections, Paun's biological model is not useful anymore to implement the living cells behavior which means that P-system of robots cannot be used to implement the living cells model.

Example and code.

Rules R
Objects w
Dissolve = false

While NOT finished
waitsync
//EVOLVE
R' = Rules_election(w,R by agent) (robot)
w' = Rules_Application (w,R,P) by agent (robot)

waitsync
//COMUNNICATION between agents (robots)
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

VII. CONCLUSIONS

This paper contributes with an implementation of a multiagent system that manages and supervises a cell membrane model. The agents along with their relations and resources are able to modify the membrane system functionality based on the agent's configuration[10]. Therefore, the main idea of this work is to define a new model created from the original membrane computing technology. This update involves a multiagent system which is the one who takes care of the process. The most interesting part of this work is that a new way to define a biological model in terms of a multiagent supervised system has been created.

It is important to stress that the membrane systems described here is a generic one. The rules election, priorities between rules, etc are not fully described as this is not the main purpose of it. Moreover, the Multiagent system technology has been generally described to understand better the concepts of systems supervised by agents.

The code provided here is a proposal but it is not the main goal of this work. By formally defining a multiagent system it would be possible to take full advantage of the Multiagent system technology and apply it into cells membrane system or any other biological model.

Refer to for checking links between Multiagents and P-systems.

Membranes agents are independent and autonomous which can modify the entire functionality of the membrane systems. The example provided shows how the membrane system can improve performance when a proper set of agents is chosen. The agents are the intelligent entities who supervise the entire membrane system and optimize the functionality.

Thus, the whole idea of this proposal is to improve and take all the possible advantages of the Multi agents systems to apply them into biological systems and make them work better in terms of performance.

The implementation of this idea has been carried out with robots. The proposal described in this section is a new model based on some of the inherent properties of P-systems plus the learning capability which is achieved by a set of autonomous robots. Although non-determinism is not a property that scenario anymore, this new scenario of P-systems & robots can obtain optimal results to known problems due to the capacity of learning. This idea is in an early stage, as there are no formal implementations of membrane computing yet. However, the idea is promising. A theoretical P-system that is able to learn by using intelligent systems such as robots, can make right decisions when applying evolution rules in P-systems' regions.

A further study is necessary to formally define the new model created by the combination of P-systems and multiagents. in particular with robots.

.

REFERENCES

[1] Paun "Computing with Membranes", Journal of Computer and System Sciences, 61(2000), and Turku Center of Computer Science-TUCS Report nº 208, 1998

[2] A. Syropoulos, E.G. Mamatas, P.C. Allilones, K.T. Sotiriades "A ["Structures and Bio-language to Simulate Transition P Systems on Digital Computers," Multiset Processing

[3] Lf. Mingo. "Self Organizing Packet Routing with Ant Colonies: Tools" Proceedings of the 11th WSEAS International Conference on AppliedComputer and Applied Computational Science (ACACOS '12)

[4] Nidhal kamel taha el-omari "Scanned document image segmentation using back-propagation artificial neural network based technique" North Atlantic University Union (NAUN) International journal of computers and communications issue 3, volume 6, 2012

[5] Wooldridge An Introduction to MultiAgent Systems" John Wiley ublications, 2002

[6] Athanasios c.. "Coordination of multiagents with a revenue-cost-sharing mechanism: a cooperative game theory approach", International journal of mathematical models and methods in Applied Sciences North Atlantic University Union (NAUN)

[7] 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).

[8] 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.

[9] "Algorithm for Application of Evolution Rules based on linear diofantic equations" Synasc 2008, Timisoara Romania September 2008.

[10] Analysys of a P-System under a Mutiagent System

perspetive, International Book Series "Information Science and Computing" pag 117 Varna Bulgaria 2009

Prof. Alberto Arteta
Dr. Arteta is currently an Assistan professor of Troy University He works in the Department 'Computer Science'. His main area of interest includes the study of mathematical models and their applications to Biology and Medical Science, and IT. He is also editor and reviewer of several prestigious international Journals.

Prof. Luis Fernando de Mingo López
Dr. Mingo is currently an Associate professor of the technical University of Madrid since 2005. He works in the Department 'Organization of data structures'. His main area of interest includes the study of patterns detections tfor data mining. He holds his Ph.D in this field and has published more then 30 papers. He is also a editor of the "International scientific society journal"

Prof. Nuria Gómez Blas
Dr. Gomez is currently an Associate professor of the technical University of Madrid since 2005. He works in the Department 'Organization of data structures'. His main area of interest includes the study of new computational models such us neural networks and Particle Swarm Optimization, and the innovation of educational method Optimization. He holds his Ph.D in this field.