FAST LINEAR ALGORITHM FOR ACTIVE RULES APPLICATION IN TRANSITION P SYSTEMS

Francisco Javier Gil, Jorge Tejedor, Luis Fernández

Abstract: Transition P systems are computational models based on basic features of biological membranes and the observation of biochemical processes. In these models, membrane contains objects multisets, which evolve according to given evolution rules. In the field of Transition P systems implementation, it has been detected the necessity to determine whichever time are going to take active evolution rules application in membranes. In addition, to have time estimations of rules application makes possible to take important decisions related to the hardware / software architectures design.

In this paper we propose a new evolution rules application algorithm oriented towards the implementation of Transition P systems. The developed algorithm is sequential and, it has a linear order complexity in the number of evolution rules. Moreover, it obtains the smaller execution times, compared with the preceding algorithms. Therefore the algorithm is very appropriate for the implementation of Transition P systems in sequential devices.

Keywords: Natural Computing, Membrane computing, Transition P System, Rules Application Algorithms

ACM Classification Keywords: D.1.m Miscellaneous – Natural Computing

Conference: The paper is selected from Sixth International Conference on Information Research and Applications – i.Tech 2008, Varna, Bulgaria, June-July 2008

Introduction

Membrane computing is a branch of natural computing witch tries to abstract computing models from the structure and the functioning of living cells. The main objective of these investigations consists of developing new computational tools for solving complex, usually conventionally-hard problems. Being more concrete, Transition P systems are introduced by Gheorghe Păun derived from basic features of biological membranes and the observation of biochemical processes [Păun, 1998]. This computing model has become, during last years, an influential framework for developing new ideas and investigations in theoretical computation.

Transition P systems are hierarchical, as the region defined by a membrane may contain other membranes. The basic components of the Transition P systems are the *membranes* that contain chemical elements (*multisets* of objects, usually represented by symbol strings) which are subject to chemical reactions (*evolution rules*) to produce other elements (another multiset). Multisets generated by evolution rules can be moved towards adjacent membranes (parent and children). This multiset transfer feeds back the system so that new multisets of symbols are consumed by further chemical reactions in the membranes.

The P system changes from a configuration to another one making a computation. Each transition or evolution step goes through two sequential steps: application of rules and communication. First, the evolution rules are applied simultaneously to the multiset in each membrane. This process is performed by all membranes at the same time. Then, also simultaneously, all the membranes communicate with their neighbors, transferring symbol multisets.

Most membrane systems are computationally universal: “P systems with simple ingredients (number of membranes, forms and sizes of rules, controls of using the rules) are Turing complete” [Păun, 2005]. This framework is extremely general, flexible, and versatile. Several classes of P systems with an enhanced parallelism are able to solve computationally hard problems (typically, NP complete problems) in a feasible time (polynomial or even linear) by making use of an exponential space.

In this paper we propose a new algorithm for evolution rules application oriented towards the implementation of Transition P systems. The developed algorithm is sequential and, it has a linear order complexity in the number of evolution rules. Moreover, it obtains the smaller execution times, compared with the preceding algorithms. Therefore, due to these characteristics, the algorithm is very appropriate for the implementation of Transition P systems in sequential devices. After this introduction, other related works appear, where the problem that is tried to solve is covered. Next are exposed the formal definitions related to the rules application in Transition P systems. Later the fast linear algorithm for rules application appears developed, finally including the comparison tests and conclusions.

Related Work

In Transition P systems, each evolution step is obtained through two consecutive phases within each membrane: in first stage the evolution rules are applied, and at the second, the communication between membranes is made. This work is centered in the first phase, the application of active rules. It exists several sequential algorithms for rules application in P systems at this moment [Ciobanu, 2002], [Fernández, 2006a] and [Tejedor, 2007], but the obtained results can be improved. In the last mentioned work is introduced an algorithm based on the elimination of active rules: this algorithm is very, interesting because is the first algorithm whose time is only limited by the number of rules, not by the objects multiset cardinality.

Additionally, in [Tejedor, 2006] is proposed a software architecture for attacking the bottleneck communication in P systems denominated “partially parallel evolution with partially parallel communications model” where several membranes are located in each processor, proxies are used to communicate with membranes located in different processors and a policy of access control to the network communications is mandatory. This obtains a certain parallelism yet in the system and an acceptable operation in the communications. In addition, it establishes a set of equations that they allow to determine in the architecture the optimum number of processors needed, the required time to execute an evolution step, the number of membranes to be located in each processor and the conditions to determine when it is best to use the distributed solution or the sequential one. Additionally it concludes that if the maximum application time used by the slowest membrane in applying its rules improves N times, the number of membranes that would be executed in a processor would be multiplied by the square root of N, the number of required processors would be divided by the same factor, and the time required to perform an evolution step would improve approximately with the same factor.

Therefore, to design software architectures it is precise to know the necessary time to execute an evolution step. For that reason, algorithms for evolution rules application that they can be executed in a delimited time are required, independently of the object multiset cardinality inside the membranes. Nevertheless, this information cannot be obtained with most of the algorithms developed until now since its execution time depends on the cardinality of the objects multiset on which the evolution rules are applied.

They have been proposed also parallel solutions -[Fernández, 2006b] and [Gil, 2007]-, but they do not obtain the required performance. The first algorithm is not completely useful, since its run time is not time delimited, and both solutions present efficiency problems due to the competitiveness between the rules, the high number of collisions with the requests and delays due to the synchronization required between processes.

Formal Definitions Related to Rules Application in P Systems

Firstly, this section formally defines the required concepts of objects multisets, evolution rules, evolution rules multiset, and applicability benchmarks (maximal and minimal) of a rule over an objects multiset. Secondly, on the basis of these definitions, requirements are specified for the new algorithm for rule evolution application.

Multisets of Objects

Definition 1: *Multiset of object. Let a finite and not empty set of objects be* O *and the set of natural numbers* N*, is defined as a multiset of object* m *as a mapping*:

$$m : \mathrm{O} \to \mathrm{N}$$
$$o \to n$$

Possible notations for a multiset of objects are:

$$m = \{(o_1, n_1), (o_2, n_2), ..., (o_m, n_m)\}$$
$$m = o_1^{n_1} \cdot o_2^{n_2} \cdot ... \cdot o_m^{n_m}$$

Definition 2: *Set of multisets of objects over a set of objects. Let a finite set of objects be* O*. The set of all the multisets that can be formed over set* O *is defined:*

$$\mathrm{M(O)} = \{m : \mathrm{O} \to \mathrm{N} \mid m \text{ is a } Multiset \text{ over } \mathrm{O}\}$$

Definition 3: *Multiplicity of object in a multiset of objects. Let an object be* $o \in O$ *and a multiset of objects* $m \in M(O)$*. The multiplicity of an object is defined over a multiset of objects such as:*

$$|\ \ |_o :\ \mathrm{O} \times \mathrm{M(O)} \to \mathrm{N}$$
$$(o, m) \to |m|_o = n \mid (o, n) \in m$$

Definition 4*: Weight or Cardinal of a multiset of objects. Let a multiset of objects be* $m \in M(O)$*. The weight or cardinal of a multiset of objects is defined as:*

$$|\ \ | :\ \mathrm{M(O)} \to \mathrm{N}$$
$$m \to |m| = \sum_{\forall o \in O} |m|_o$$

Definition 5: *Multiset support. Let a multiset of objects be* $m \in M(O)$ *and* $P(O)$ *the power set of* O*. The support for this multiset is defined as:*

$$Supp :\ \mathrm{M(O)} \to P(\mathrm{O})$$
$$m \to Supp(m) = \{ o \in \mathrm{O} \ /\ |m|_o > 0 \}$$

Definition 6: *Empty multiset. This is the multiset represented by* $\varnothing_{M(O)}$ *and which satisfies:*

$$\varnothing_{\mathrm{M(O)}} \Leftrightarrow |m| = 0 \Leftrightarrow Supp(m) = \varnothing$$

Definition 7: *Inclusion of multisets of objects. Let two multisets of objects be* $m_1, m_2 \in M(O)$*. The inclusion of multisets of objects is defined as:*

$$m_1 \subset m_2 \Leftrightarrow |m_1|_o \le |m_2|_o \quad \forall o \in \mathrm{O}$$

Definition 8*: Sum of multisets of objects. Let two multisets of objects be* $m_1, m_2 \in M(O)$*. The sum of multisets of objects is defined as:*

$$+ :\ \mathrm{M(O)} \times \mathrm{M(O)} \ \to\ \mathrm{M(O)}$$
$$(m_1, m_2) \ \to\ \{(o, |m_1|_o + |m_2|_o)\ \ \forall o \in \mathrm{O}\}$$

Definition 9*: Subtraction of multisets of objects. Let two multisets of objects be* $m_1, m_2 \in M(O)$*, and* $m_2 \subset m_1$*. The subtraction of the multisets of objects is defined as:*

$$- :\ \mathrm{M(O)} \times \mathrm{M(O)} \ \to\ \mathrm{M(O)}$$
$$(m_1, m_2) \ \to\ \{(o, |m_1|_o - |m_2|_o)\ \ \forall o \in \mathrm{O}\}$$

Definition 10: *Intersection of multisets of objects. Let two multisets of objects be* $m_1, m_2 \in M(O)$*. The intersection of multisets of objects is defined as:*

$$\bigcap : \mathrm{M(O)} \times \mathrm{M(O)} \to \mathrm{M(O)}$$
$$(m_1, m_2) \ \to m_1 \bigcap m_2 = \{(o, \min(|m_1|_o, |m_2|_o)\ \ \forall o \in \mathrm{O}\}$$

Definition 11*: Scalar product of multiset of objects by a natural number. Let a multiset be* $m_2 \in M(O)$ *and a natural number* $n \in N$. *The scalar product is defined as:*

$$\cdot : M(O) \times N \to M(O)$$
$$(m, n) \to m \cdot n = \{(o, | m |_o \cdot n) \quad \forall o \in O\}$$

Evolution Rules

Definition 12*: Evolution rule over a set of objects with target in* T *and with no dissolution capacity. Let a set of objects be* O, $a \in M(O)$ *a multiset over* O, $T = \{here, out\} \cup \{in_j / 1 \le j \le p\}$ *a set of targets and* $c \in M(O \times T)$ *a multiset over* $O \times T$. *An evolution rule is defined like a tuple*:

$$r = (a, c)$$

Definition 13: *Set of evolution rules over a set of objects and targets in* T. *This set is defined as:*

$$R(O, T) = \{r \mid r \text{ is a rule over } O \text{ and } T\}$$

Definition 14: *Antecedent of Evolution Rule. Let an evolution rule be* $r \in R(O, T)$. *The antecedent of an evolution rule is defined over a set of objects as:*

$$input : R(O, T) \to M(O)$$
$$(a, c) \to input(r) = a \mid r = (a, c) \in R(O, T)$$

Definition 15: *Evolution rule applicable over a multiset of objects. Let an evolution rule be* $r \in R(O, T)$ *and a multiset of objects* $m \in M(O)$*, it is said that an evolution rule is applicable over a objects multiset if and only if:*

$$\Delta_r(m) \Leftrightarrow input(r) \subset m$$

Definition 16: *Set of evolution rules applicable to a multiset of objects. Let a set of evolution rules be* $R \in P(R(O, T))$ *and a multiset of objects* $m \in M(O)$. *The set of evolution rules applicable to a multiset of objects is defined as:*

$$\Delta^* : P(R(O, T)) \times M(O) \to P(R(O, T))$$
$$(R, m) \to \Delta^*_R(m) = \{r \in R \mid \Delta_R(m) = true\}$$

Property 1: *Maximal applicability benchmark of evolution rule over a multiset of objects*. Let an evolution rule be $r \in R(O, T)$ and a multiset of objects $m \in M(O)$. The maximal applicability benchmark of a rule in a multiset is defined as:

$$\Delta\lceil \ \rceil : R(O, T) \times M(O) \to N$$
$$(r, m) \to \Delta_r\lceil m \rceil = \min\left\{ \frac{| m |_o}{| input(r) |_o} \ \forall o \in Supp(m) \wedge \left| input(r) \right|_o \neq 0 \right\}$$

Property 2: *Minimal applicability benchmark of evolution rule over a multiset of objects and a set of evolution rules*. Let an evolution rule be $r \in R(O, T)$, a multiset of objects $m \in M(O)$ and a set of evolution rules $R \in P(R(O, T))$. The minimal applicability benchmark is defined as the function:

$$\Delta\lfloor \ \rfloor : R(O, T) \times M(O) \times P(R(O, T)) \to N$$
$$(r, m, R) \to \Delta_r\lfloor m \rfloor = \Delta_r \left\lceil m - \left(m \cap \sum_{\forall ri \in R - \{r\}} input(r_i) \cdot \Delta_{ri}\lceil m \rceil \right) \right\rceil$$

Property 3: *An evolution rule* $r \in R(O, T)$ *is applicable to a multiset of objects* $m \in M(O)$ if and only if the maximal applicability benchmark is greater or equal to 1.

$$\Delta_r(m) \Leftrightarrow \Delta_r\lceil m \rceil \ge 1$$

Property 4: The maximal applicability benchmark of a rule $r \in R(O, T)$ *over an object multiset* $m \in M(O)$ *is greater than or equal to the maximal applicability benchmark of the rule in a subset of the object multiset.*

$$\Delta_r\lceil m_1 \rceil \geq \Delta_r\lceil m_2 \rceil \ \ \forall m_1, m_2 \in M(O) \ \ | m2 \subset m1$$

Property 5: If the maximal applicability benchmark of a rule $r \in R(O, T)$ over a multiset of objects $m \in M(O)$ is 0, then the maximal applicability benchmark of the rule r over the sum of $input(r)$ and m is equal to the maximal applicability benchmark of the $input(r)$ and equal to 1.

$$\Delta_r\lceil m \rceil = 0 \Rightarrow \Delta_r\lceil input(r) + m \rceil = \Delta_r\lceil input(r) \rceil = 1$$

Multisets of Evolution Rules

Definition 17: *Multiset of evolution rules*. Let a finite and not empty set of evolution rules be $R(O, T)$ and the set of natural numbers N, a multiset of evolution rules is defined as the mapping:

$$M_{R(O,T)} : R(O,T) \to N$$
$$r \to n$$

All definitions related to multisets of objects can be extended to multisets of rules.

Definition 18: *Linearization of evolution multiset of rules*. Let a multiset of evolution rules be $m_R = r_1^{k1} \cdot r_2^{k2} \cdot \ldots \cdot r_q^{kq} \in M_{R(O,T)}$ linearization of m_R is defined as:

$$\sum_{i=1}^{q} r_i \cdot k_i \in R(O, T)$$

Requirements of Application of Evolution Rules over Multiset of objects

Application of evolution rules in each membrane of P Systems involves subtracting objects from the objects multiset by using rules antecedents. Rules used are chosen in a non-deterministic manner. The process ends when no rule is applicable. In short, rules application to a multiset of object in a membrane is a process of information transformation with input, output and conditions for making the transformation.

Given an object set $O = \{o_1, o_2, ..., o_m\}$ where $m > 0$, the input to the transformation process is composed of a multiset $\omega \in M(O)$ and $R \in R(O, T)$, where:

$$\omega = o_1^{n_1} \cdot o_2^{n_2} \cdot \ldots \cdot o_m^{n_m}$$
$$R = \{r_1, r_2, ..., r_q\} being\ q > 0$$

In fact, the transformation only needs rules antecedents because this is the part that acts on ω. Let these antecedents be:

$$input(r_1) = o_1^{n_1^i} \cdot o_2^{n_2^i} \cdot \ldots \cdot o_m^{n_m^i} \quad \forall i = \{1, 2, ..., q\}$$

The output of the transformation process will be a objects multiset of $\omega' \in M(O)$ together with the multiset of evolution rules applied $\omega_R \in M_{R(O,T)}$.

$$\omega' = o_1^{n'_1} \cdot o_2^{n'_2} \cdot \ldots \cdot o_m^{n'_m}$$
$$\omega_R = r_1^{k_1} \cdot r_2^{k_2} \cdot \ldots \cdot r_q^{k_q}$$

Conditions for making the transformation are defined according to the following requirements:

Requirement 1: The transformation process is described through the following system of equations:

$$n_1 = n_1^1 \cdot k_1 + n_1^2 \cdot k_2 + ... + n_1^q \cdot k_q + n_1^{'}$$
$$n_2 = n_2^1 \cdot k_1 + n_2^2 \cdot k_2 + ... + n_2^q \cdot k_q + n_2^{'}$$
$$...$$
$$n_m = n_m^1 \cdot k_1 + n_m^2 \cdot k_2 + ... + n_m^q \cdot k_q + n_m^{'}$$

That is:

$$\sum_{j=1}^{q} n_i^j \cdot k_j + n_i^{'} = n_i \quad \forall i = \{1, 2, ..., m\}$$

or

$$\sum_{i=1}^{q} input(r_i) \cdot k_i + \omega' = \omega$$

The number of equations in the system is the cardinal of the set O. The number of unknowns in the system is the sum of the cardinals of the set O and the number of rules of R. Thus, the solutions are in this form:

$$(n_1^{'}, n_2^{'}, ..., n_m^{'}, k_1, k_2, ..., k_q) \in \mathrm{N}^{m+q}$$

Meeting the following restrictions:

$$0 \le n_i^{'} \le n_i \quad \forall i = \{1, 2, ..., m\}$$

Moreover, taking into account the maximal and minimal applicability benchmarks of each rule, the solution must satisfy the following system of inequalities:

$$\Delta_{r_j} \lfloor \omega \rfloor \le k_j \le \Delta_{r_j} \lceil \omega \rceil \quad \forall j = \{1, 2, ..., q\}$$

Requirement 2: No rule of the set R can be applied over the multiset of objects ω', that is:

$$\Delta_r(\omega') = false \quad \forall r \in \mathrm{R}$$

Having established the above requirements, the system of equations may be incompatible (no rule can be applied) determinate compatible (there is a single multiset of rules as the solution to the problem) or indeterminate compatible (there are many solutions). In the last case, the rule application algorithm must provide a solution that is randomly selected from all possible solutions in order to guarantee non-determinism inherent to P systems.

Fast Linear Algorithm for Active Rules Application in Transition P Systems

This section describes the fast linear algorithm for active rules application to a multiset of objects whose execution time depends on the number of rules. The initial input is a set of active evolution rules for the corresponding membrane -the rules are applicable and useful- and the initial membrane multiset of objects. The final results are the complete multiset of applied evolution rules and the obtained multiset of objects after rules application.

The algorithm is based on the one by one elimination of rules: when a rule has been applied to its maximal applicability benchmark, this rule lets be active, and therefore it is eliminated. The algorithm finishes when all rules have been eliminated. The algorithm is made up of two phases:

1. At the first phase all rules belonging to the set of active rules -except one- are applied a random number of times between 0 and its maximal applicability benchmark. In this way, each active rule has a possibility of being applied.

2. In the second phase, all the rules -beginning by the one excluded of the previous phase- are applied to its maximal applicability benchmark. Consequently, there it is not left any rule applicable, and the algorithm finishes generating like result the multiset of rules applied and the final multiset of objects.

In order to facilitate the explanation of the algorithm, the set of initially active rules is represented like an ordered sequence R and an auxiliary structure called Active. The position of any rule r_i in the sequence is i. Active[i] indicates if the rule r_i continues active. The rule excluded in first stage is the one that is in the last position of the sequence (it can be any rule of the set of active rules). The pseudocode of the algorithm is as follows:

```
( 1)  ω' ← ω
( 2)  ωR ← ØMR(U)
( 3)  FOR i = 1 TO |R| - 1 DO // Phase 1
( 4)      BEGIN
( 5)      Max ← ΔR[i](ω')
( 6)      IF ( Max ≠ 0 ) THEN
( 7)          BEGIN
( 8)          K ← random(0, Max)
( 9)          ωR ← ωR + {R[i]^K}
(10)          ω' ← ω' − input(R[i]) · K
(11)          Active[i] = (K < Max)
(12)          END
(13)      ELSE Active[i] = false
(14)      END
(15)
(16)  Active[|R|] = true
(17)  FOR i = |R| DOWNTO 1 DO // Phase 2
(18)      IF ( Active[i] ) THEN
(19)          BEGIN
(20)          Max ← ΔR[i](ω')
(21)          ωR ← ωR + {R[i]^Max}
(22)          ω' ← ω' − input(R[i]) · Max
(23)          END
```

As it has been previously indicated, the algorithm is made up of two phases. In first stage is offered the possibility to all the rules -except one- to be applied between 0 and their maximum applicability benchmark. In addition is determined if a rule lets be active. Rules can let be active in this stage due to two possible reasons: a) the rule has been applied to its maximum applicability, or b) other preceding rules have consumed the necessary objects so that the rule can be applied.

The second phase begins supposing like active the last rule (observe that this is not necessarily certain). Next, beginning by the one excluded of the first phase, all the supposedly active rules are applied to its maximum applicability. After this step all the rules let be active, and the application algorithm finished their execution. As it can be seen, the algorithm executes a finite and well-known number of operations, which only depends on the initial number of active rules.

In the next sections we are going to demonstrate the correctness of the exposed algorithm, as well as the efficiency analysis.

Algorithm Correctness

The presented algorithm is correct because:

Lemma 1: *The algorithm is finite.*

Proof: The first two lines are basic operations. The first loop -from line (3) to (14)- is exactly executed $|R| - 1$ times, and its body only contains simple operations. The second loop -from line (16) to (23)- is exactly executed $|R|$ times, and also its body only contains simple operations.

Lemma 2: *No evolution rule is applicable to ω'.*

Proof: The sequence R initially contains all rules applicable to *ω'*. Owing to property 3 we know that a rule with a maximal applicability benchmark equal to zero is not applicable. After the execution of the second phase of the algorithm, the maximal applicability of all the rules is zero. Therefore, at the end of the algorithm execution, it is not left any rule applicable to *ω'*.

Lemma 3: *Any result generated is a possible solution.*

Proof: The multiset of rules applied ω_R is obtained by the multiple applications of the active rules in both phases. In addition, since in the second phase each active rule is applied to its maximal applicability benchmark, after the execution of the algorithm no rule is applicable over *ω'* (requirement 2), and the result generated is a possible solution.

Lemma 4: *Any solution possible is generated by the algorithm*

Proof: Phase 1 of the algorithm -from line (3) to (14)- guarantees that any possible solution can be generated. It is enough whereupon the appropriate number is generated in line 8, when the number of applications of a rule is determined. In the second phase it would be only needed to apply the last rule the appropriate number of times.

Lemma 5: *The algorithm is not determinist*

Proof: This occurs when a rule is not the last one in the set, it is applied a randomly determined number of times (sentence 8) between zero and its maximal applicability value.

Efficiency Analysis

Examining the algorithm it is possible to observe that in the two phases, the heaviest operations are those for calculating the maximal applicability benchmark (sentences 5 and 20), the scalar product of the *input* of a rule by a whole number and the difference of two multisets (sentences 10 and 22). These operations are made in both phases in the worse case. All these operations are linearly dependant on the cardinal of the *multiset support* ω.

$$\#operations\_per\_iteration \approx 3 \cdot Supp(\omega)$$

Moreover, the worst case of the fast linear algorithm occurs when sentences 6 and 11 are evaluated always affirmatively, or what is the same, when no rule becomes inactive after the execution of first stage. In this case, there is no improvement in the behavior of the algorithm and the number of iterations executed is:

$$\#iterations = (|R| - 1) + |R| = 2 \cdot |R| - 1$$

Therefore, the number of operations executed at worst case by the algorithm is:

$$\#operations = (2 \cdot |R| - 1) \cdot 3 \cdot Supp(\omega)$$

So the execution time of the algorithm at worst is linear dependant of the number of rules.

Comparison Tests

The experimental tests have compared the execution time of the *Fast Linear algorithm* (FLA) with the one that was fastest until now, that is *Active Rules Elimination* (ARE) algorithm [Tejedor, 2007]. The experimental trial game used to test both algorithms has taken into account 3 parameters:

1. *Number of objects of the multiset*. In comparative the value of this parameter is 16
2. *Number of rules (q)*. The value of q has taken all the values from the set {1, 2, 4, 8, 16, 32, 64, 128}
3. *Relationship between the cardinal of the multiset and the cardinal of the sum of inputs of the active rules set (r).* The value of r has taken all the values from the set {1, 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7}

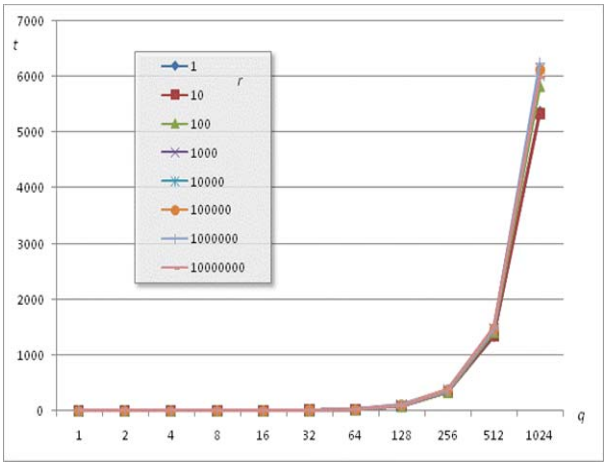

Figure 1.- Evolution of the execution times difference

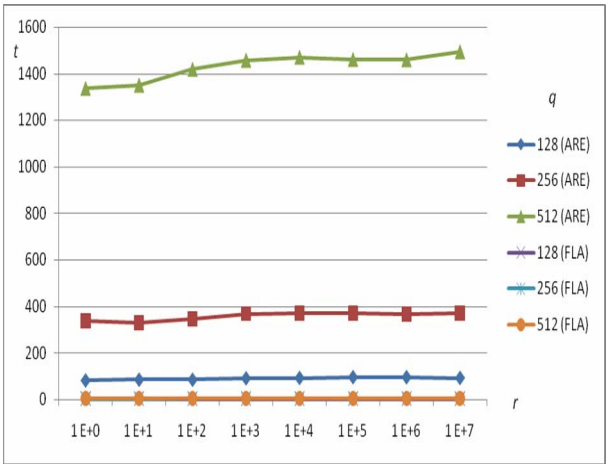

Figure 2.- Execution time of ARE and FLA

Figure 1 shows a graphic with the evolution of the execution times difference obtained in the tests between the REA and the FLA. Each curve of the graphic represents the difference of execution time of the ARE algorithm with regards to the execution time of FLA algorithm for each of the values of the relationship of the cardinals (r). In this graphic you can see that FLA algorithm is always better than ARE algorithm independently of the number of rules and the relationship between cardinals.

In Figure 2 it can be observed that the execution time grows modestly -with both algorithms- with the value of r. The parameter that influences more in the results is the number of rules (q). The obtained results are logical considering that the complexity of the ARE algorithm is order of square of q, and the complexity of the FLA is q linear.

Conclusions

This paper introduces a new algorithm for active rules application to a multiset of objects based on rules elimination in transition P systems. This algorithm attains a certain degree of parallelism, as a rule can be applied a great number of times in a single step. The number of operations executed by the algorithm is time delimited, because it only depends on the number of rules of the membrane. The number of rules of the membrane is well known static information studying the P system, thus allowing determining beforehand the algorithm execution time. This information is essential to calculate the number of membranes that have to be located in each processor in distributed implementation architectures of P systems to achieve optimal times with minimal resources.

We think that the presented algorithm can represent an important contribution in particular for the problem of the application of rules in membranes, because it presents high productivity and it allows estimate the necessary time to execute an evolution step. Additionally, this last one allows making important decisions related to the implementation of P systems, like the related ones to the software architecture.

Bibliography

[Ciobanu, 2002] G. Ciobanu, D. Paraschiv, "*Membrane Software. A P System Simulator*". Pre-Proceedings of Workshop on Membrane Computing, Curtea de Arges, Romania, August 2001, Technical Report 17/01 of Research Group on Mathematical Linguistics, Rovira i Virgili University, Tarragona, Spain, 2001, 45-50 and Fundamenta Informaticae, vol 49, 1-3, 61-66, 2002.

[Ciobanu, 2006] G. Ciobanu, M. Pérez-Jiménez, Gh. Păun, "*Applications of Membrane Computing*". Natural Computing Series, Springer Verlag, October 2006.

[Fernández, 2006a] Fernández, L. Arroyo, F. Castellanos, J. et al (2006) "*New Algorithms for Application of Evolution Rules based on Applicability Benchmarks*". BIOCOMP 06, Las Vegas (USA)

[Fernández, 2006b] L. Fernández, F. Arroyo, J. Tejedor, J. Castellanos. "*Massively Parallel Algorithm for Evolution Rules Application in Transition P System*". Seventh Workshop on Membrane Computing, WMC7, Leiden (The Netherlands). July, 2006

[Gil, 2007] Gil, F. J. Fernández, L. Arroyo, F. et al "*Delimited Massively Parallel Algorithm based on Rules Elimination for Application of Active Rules in Transition P Systems*" i.TECH-2007. Varna (Bulgaria).

[Păun, 1998] G. Păun. "*Computing with Membranes*". In: Journal of Computer and System Sciences, 61(2000), and Turku Center of Computer Science-TUCS Report nº 208, 1998.

[Păun, 2005] G. Păun. "*Membrane computing. Basic ideas, results, applications*". In: Pre-Proceedings of First International Workshop on Theory and Application of P Systems, Timisoara (Romania), pp. 1-8, September, 2005.

[Tejedor, 2006] J. Tejedor, L. Fernández, F. Arroyo, G. Bravo. "*An Architecture for Attacking the Bottleneck Communications in P systems*". In: Artificial Life and Robotics (AROB 07). Beppu (Japan), January 2007.

[Tejedor, 2007] J. Tejedor, L. Fernández, F. Arroyo, A. Gutiérrez. "*Algorithm of Active Rules Elimination for Evolution Rules Application*". In 8th WSEAS Int. Conf. on Automation and Information, Vancouver (Canada), June 2007.

Authors' Information

F. Javier Gil Rubio – Dpto. de Organización y Estructura de la Información, E.U. de Informática.
Natural Computing Group, Universidad Politécnica de Madrid, Spain; e-mail: jgil@eui.upm.es

Jorge A. Tejedor Cerbel - Dpto. de Organización y Estructura de la Información, E.U. de Informática.
Natural Computing Group, Universidad Politécnica de Madrid, Spain; e-mail: jtejedor@eui.upm.es

Luis Fernández Muñoz - Dpto. de Lenguajes, Proyectos y Sistemas Informáticos, E.U. de Informática.
Natural Computing Group, Universidad Politécnica de Madrid, Spain; e-mail: setillo@eui.upm.es