Krassimir Markov, Vitalii Velychko,
Lius Fernando de Mingo Lopez, Juan Casellanos
(editors)

New Trends in Information Technologies

I T H E A
SOFIA
2010

Krassimir Markov, Vitalii Velychko, Lius Fernando de Mingo Lopez, Juan Casellanos (ed.)

New Trends in Information Technologies

ITHEA®

Sofia, Bulgaria, 2010

ISBN 978-954-16-0044-9

First edition

Recommended for publication by The Scientific Concil of the Institute of Information Theories and Applications FOI ITHEA

This book maintains articles on actual problems of research and application of information technologies, especially the new approaches, models, algorithms and methods of membrane computing and transition P systems; decision support systems; discrete mathematics; problems of the interdisciplinary knowledge domain including informatics, computer science, control theory, and IT applications; information security; disaster risk assessment, based on heterogeneous information (from satellites and in-situ data, and modelling data); timely and reliable detection, estimation, and forecast of risk factors and, on this basis, on timely elimination of the causes of abnormal situations before failures and other undesirable consequences occur; models of mind, cognizers; computer virtual reality; virtual laboratories for computer-aided design; open social info-educational platforms; multimedia digital libraries and digital collections representing the European cultural and historical heritage; recognition of the similarities in architectures and power profiles of different types of arrays, adaptation of methods developed for one on others and component sharing when several arrays are embedded in the same system and mutually operated.

It is represented that book articles will be interesting for experts in the field of information technologies as well as for practical users.

General Sponsor: Consortium FOI Bulgaria (www.foibg.com).

Printed in Bulgaria

ISBN 978-954-16-0044-9

C\o Jusautor, Sofia, 2010

A BOUNDED ALGORITHM BASED ON APPLICABILITY DOMAINS FOR THE APPLICATION OF ACTIVE RULES IN TRANSITION P-SYSTEMS

F. Javier Gil, Jorge A. Tejedor, Luis Fernández

***Abstract**: Transition P systems are a computational model based on the basic features of biological membranes and in the observation of biochemical processes. In this model, a membrane contains object multisets, which evolve following a determine set of evolution rules. The system changes from an initial configuration to another one performing a computation by applying these rules in a non-deterministic maximally parallel way.*

Previous works about the rule application algorithms follow a common pattern: to choose a rule applicable on the objects multiset, and to apply that rule a number of times. This pattern is repeated until there is no applicable rule. In this paper, we present an algorithm that proposes a new approach that consists of the following steps: (i) To analyze the antecedent rules to establish the applicability domains; (ii) to select some applicable rules depending on the objects multiset available inside the membrane. Unlike previous algorithms, this new approach always selects an applicable rule in each iteration, avoiding iterations with non-applicable rules. In addition, this is a bounded algorithm, which is a necessary condition that facilitates decision-making in membrane architecture design.

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

***ACM Classification Keywords**: D. Software. D.1 Programming Techniques*

Introduction

Membrane computing is a branch of natural computing witch tries to abstract computing models from the structure and the functioning of biological cells, particular of the cellular membranes. 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.

An essential ingredient of a P system is its membrane structure. 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 nondeterministic maximally parallel application of rules throughout the system is a transition between system states, and a sequence of transitions is called a computation. Each transition or *evolution step* goes through two sequential phases: the application of the evolution rules and communication. First, the evolution rules are applied simultaneously to the object 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 the application of evolution rules oriented towards the implementation of Transition P systems. The proposed algorithm has a linear order complexity in the number of evolution rules and provides an optimization of execution time compared to preceding contributions. Due to these characteristics, the algorithm is very appropriate for the implementation of Transition P systems in sequential devices.

After this introduction, related works are presented, where the problem that is tried to solve is covered. Then are exposed the formal definitions related to the application of rules in Transition P systems. Later appears developed the bounded algorithm based on applicability domains, including finally the efficiency analysis, and the conclusions.

Related Work

In Transition P systems, each evolution step is obtained through two consecutive phases within each membrane: in the first stage the evolution rules are applied, and at the second is made the communication between membranes. This work is centered in the first phase, the application of active rules. It exists at this moment several sequential algorithms for rules application in P systems [Ciobanu, 2002], [Fernández, 2006a], [Gil, 2008] and [Tejedor, 2007], but the performance of these algorithms 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 execution 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.

Finally, the algorithm proposed in [Gil, 2008] obtains a good performance, but these can be improved in certain situations (specifically, when the relationship between the number of rules and the number of objects is high). Analyzing the behavior of the latter shows that this algorithm performs several iterations in which objects are not consumed due to the competitiveness between the rules. The bounded algorithm based on applicability domains is designed to avoid these unnecessary iterations, thus obtaining better performance.

Formal definitions related to rules application in P systems

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

Multiset of Objects

Definition 1: Multiset of objects. 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:

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

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:

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

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

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

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

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

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

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

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

$$m_1 \subset m_2 \Leftrightarrow |m_1|_o \leq |m_2|_o \quad \forall o \in O$$

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:

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

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:

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

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:

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

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:

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

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, $T = \{here, out\} \cup \{in_j / 1 \leq j \leq p\}$ 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:

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

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:

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

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

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

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

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

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

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

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

$$\Delta_r\lceil m1 \rceil \geq \Delta_r\lceil m2 \rceil \ \ \forall m1, m2 \in M(\mathrm{O}) \ \mid m2 \subset m1$$

Property 5: If the maximal applicability benchmark of a rule $\mathrm{r} \in \mathrm{R}(\mathrm{O}, \mathrm{T})$ over a multiset of objects $\mathrm{m} \in \mathrm{M}(\mathrm{O})$ is 0, then the maximal applicability benchmark of the rule r over the sum of $\mathrm{input}(\mathrm{r})$ and m is equal to the maximal applicability benchmark of the $\mathrm{input}(\mathrm{r})$ 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 $\mathrm{R}(\mathrm{O}, \mathrm{T})$ and the set of natural numbers N, a multiset of evolution rules is defined as the mapping:

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

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 $\mathrm{m_R} = \mathrm{r_1}^{\mathrm{k1}} \cdot \mathrm{r_2}^{\mathrm{k2}} \cdot \ldots \cdot \mathrm{r_q}^{\mathrm{kq}} \in \mathrm{M_{R(O, T)}}$ linearization of $\mathrm{m_R}$ is defined as:

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

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 $\omega_R \in M_{R(O, T)}$.

$$\omega' = o_1^{n'_1} \cdot o_2^{n'_2} \cdot \ldots \cdot o_m^{n'_m}$$

$$\omega_R = r_1^{k_1} \cdot r_2^{k_2} \cdot \ldots \cdot r_q^{k_q}$$

The conditions to perform the transformation are defined according to the following requirements:

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

$$n_1 = n_1^1 \cdot k_1 + n_1^2 \cdot k_2 + \ldots + n_1^q \cdot k_q + n_1'$$

$$n_2 = n_2^1 \cdot k_1 + n_2^2 \cdot k_2 + \ldots + n_2^q \cdot k_q + n_2'$$

$$\ldots$$

$$n_m = n_m^1 \cdot k_1 + n_m^2 \cdot k_2 + \ldots + n_m^q \cdot k_q + n_m'$$

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:

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

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 the non-determinism inherent to P systems.

Bounded Algorithm based on Applicability Domains for the Application of Active Rules in Transition P Systems

This section describes the bounded algorithm based on applicability domains the application of active 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 multiset of objects obtained after the application of rules.

The Rule Applicability Domains

The applicability domains determination consist in a preliminary study of the antecedents of all the evolution rules, determining at each moment the set of rules that can be applied depending on the object multisets. Thus, at any time since it is known the object multisets in the membrane, it is possible to know exactly which set of rules that can be applied. Moreover, the computing of the applicability domains may be made prior to the process of application of rules, and therefore before the P system begins to evolve.

In addition, since at this point of transition between states of the P system only objects are consumed, the set of rules are reduced with each iteration of the algorithm, eliminating those rules that after each iteration are no longer applicable.

The Rule Application Algorithm

Before the application of the evolution rules, the domains of applicability are calculated. The algorithm is made up of two phases:

At the first phase an applicable rule set is randomly selected, and then applied a random number of times between one and its maximal applicability benchmark. The selected rule set is not reapplied during this first phase.

In the second phase all the applicable rule sets are applied to its maximal applicability benchmark until there is no applicable rule. Consequently, when this phase ends 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 object called *applicableRuleSet*. This object has been built upon the study of the domains of applicability, and contains all sets of rules applicable at a given moment. The algorithm pseudo code is as follows:

```
(01) ω' ← ω;
(02) ω_R ← Ø_MR(U);
(03) applicableRuleSet.init(R[i], ω);
(04) do {                                          // Phase 1
(05)     r ← applicableRuleSet.getRandom();
(06)     K ← random(1, Δ_R[r]⌈ω'⌉);
(07)     ω_R ← ω_R + {R[r]^K};
(08)     ω' ← ω' − K × input(R[r]);
(09)     applicableRuleSet.remove(r);
(10) } while (!applicableRuleSet.empty());
(11)
(12) applicableRuleSet.init(R[i], ω');            // Phase 2
(13) while (!applicableRuleSet.empty()) {
(14)     i ← applicableRuleSet.getNext();
(15)     ma ← Δ_R[i]⌈ω'⌉;
(16)     ω_R ← ω_R + {R[i]^ma};
(17)     ω' ← ω' − ma × input(R[i]);
(18)     applicableRuleSet.remove(i);
(19) }
```

As it has been previously indicated, the algorithm is made up of two phases. In first stage is offered the possibility to all the applicable set of rules to be applied between one and their maximum applicability benchmark. The rule sets can let be active in this stage due to two possible reasons: a) the rule set has been applied to its maximum applicability, or b) other precedents rule sets have consumed the necessary objects so that the rule set can be applied.

At the beginning of the second phase the applicable set of rules is recalculated. Then in a loop are selected and applied to its maximum applicability each applicable rule sets, until there is no applicable rule and the application algorithm finish their execution. Through the use of the domains of applicability, all iterations consume objects. As seen, the algorithm executes a finite and bounded number of operations.

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 algorithm is composed of basic operations and two loops. In these loops always reduces the number of objects in the membrane, and therefore the algorithm terminates when there is no applicable rule.

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

Proof: 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 applicable rule 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, the second phase ends when there are no applicable rules 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 (4) to (10) - guarantees that any possible solution can be generated. It is enough whereupon the appropriate number is generated in line 6, 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: The algorithm is nondeterministic since from the same initial input is able to generate any possible solution (multiset of rules that make the algorithm finishes). In phase 1, both the selection of the set of rules, as the choice of the number of times they apply, are made randomly.

Efficiency Analysis

Examining the algorithm it is possible to observe that in the two phases, the heaviest operations are those that calculate the maximal applicability benchmark (sentences 6 and 15), the scalar product of the *input* of a rule by a whole number and the difference of two multisets (sentences 08 and 17). 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, examining the first phase, the worst case of the algorithm occurs when - in sentence 05 – is selected a set that contains only one rule. As the rule is eliminated from the set of applicable rules, the first loop will execute at most R times. The same is true in the second phase, bringing that the number of iterations of both loops will be:

$$\#iterations = 2 \cdot |\,\mathrm{R}\,|$$

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

$$\#operations = (\,2 \cdot |\,\mathrm{R}\,|\,) \times 3 \cdot Supp(\omega)$$

So the execution time of the algorithm at worst is bounded and linear dependant of the number of rules. Moreover, in practice, it is expected that the number of iterations of both loops is smaller, and therefore the efficiency will be higher.

Conclusions

This paper introduces a new algorithm for active rules application to a multiset of objects based on applicability domains in transition P systems. This algorithm attains a certain degree of parallelism, as a set of rules can be applied a great number of times in a single step. The number of operations executed by the algorithm is bounded, 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 the determination prior to the application of the rules. 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).

[Gil, 2008] Gil, F. J. Tejedor, J. A. Fernández, "Fast Linear Algorithm for Active Rules Application in Transition P Systems" i.TECH-2008. 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*