Calculating maximal multisets of objects by using RAM as support

Alberto Arteta, Fernando Arroyo, Ángel Goñi

Natural computing group (UPM)
Ctra. de Valencia, Km. 7
28031 - MADRID
Phone.: 913367903-04

aarteta@eui.upm.es , farroyo@eui.upm.es, agmoreno@gcn.upm.es

***Abstract*:** Within the membrane computing research field, there are many papers about software simulations and a few about hardware implementations. In both cases, algorithms are implemented. These algorithms implement membrane systems in software and hardware that try to take advantages of maximal parallelism. P-systems are parallel and non deterministic systems which simulate membranes behavior when processing information.
This paper describes the evolution rules application process and it presents software techniques for calculating maximal multisets on every evolutionary step. These techniques improve the best performance achieved when applying evolution rules over multisets of objects.

Keywords: P-systems, Parallel systems, Natural Computing, evolution rules application, structure.

I. INTRODUCTION

Membrane computing is a parallel and distributed computational model based on the membrane structure of living cells [1]. P-systems are the structures that reproduce the information processing occurring in living cells. Nowadays, it can be found that several P Systems simulators are much elaborated" [2].At this point, there is a problem with the parallelism synthesized in [2] by: "parallelism -a dream of computer science, a common sense in biology". This is the reason why, Păun avoids "to plainly saying that we have 'implementations of P systems', because of the inherent non-determinism and the maximal parallelism of the basic model, features which cannot be implemented, at least in principle, on the usual electronic computer. […]

There are several uses of the p-systems. P-systems can be used to increase performance when dealing with known problems; for example, the *knapsack problem* [3].

II. WORK RESSUME

We are going to define concepts regarding membrane computing as , p-systems, multisets of objects, multiplicities, etc; then we will describe the evolution rules application phase. After this, we will create a *n-dimensional structure* throughout an application that establishes a link between the *initial multisets*, and the number of times that each *evolution rule* should be applied in order to obtain a maximal *multiset.* The number of entries this structure has will be determined by a prefixed value called *"benchmark"* .Finally, some conclusions will be established. These conclusions will be the result of doing a study in detail throughout the entire work presented here.

III. DEFINITIONS

Definition 3.1 Transition P Systems

A Transition P System of degree n, n > 1 is a construct
$= $ [illegible]

Where: V is an alphabet; its elements are called objects; μ is a membrane structure of degree n, with membranes and regions labeled in a one-to-one manner with elements in a given set ; we always use labels 1,2,..,n;

$\omega_i\ 1\le i\le n$, are strings from V^* representing multisets over V associated with the regions 1,2,..,n of μ

$R_i\ 1\le i\le n$, 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 $u\to v$ i_o is a number between 1 and n which specifies the output membrane of Π

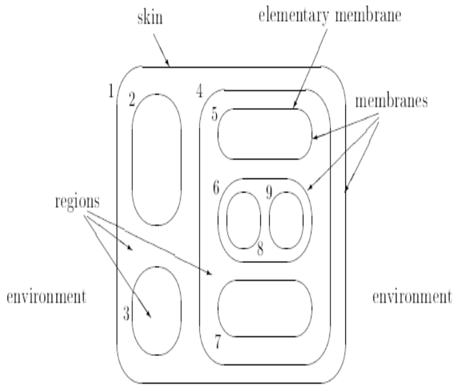

Fig 1 P-system structure

Definition 3.2 *Multiset of objects*

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

$$M : U \rightarrow N$$
$$a_i \rightarrow u_1$$

Where a_i is an object and u_i its multiplicity. Here are some representations:

[illegible]

Note: *Initial Multiset* is the multiset existing within a given region in where no application of evolution rules has occurred yet.

Definition 3.3
Evolution rule with objects in *U* and targets in *T* is defined by $r = (m, c, \delta)$ where $m \in M(U), c \in M(UxT)\, and\, \delta \in \{to\ dissolve, not\ to\ dissolve\}$

From now on 'c' will be referred a s the consequent of the evolution rule 'r'

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

Definition 3.4 *Multiplicity of an object in a multiset of objects M (U)*
Let $a_i \in U$ be an object and let $m \in M(U)$ be a multiset of objects. The multiplicity of an object is defined over a multiset of objects such as:

$$| \ |_{a_i} : U \times M(U) \rightarrow N$$
$$(a_i, m) \rightarrow |m|_{a_i} = M(a_i) = u_i$$

Definition 3.4.1 *Multiplicity of an object in an evolution rule r*
Let $a_i \in U$ be an object and let $R(U,T)$ be a multiset of evolution rules. Let $r = (m, c, \delta) \in R(U,T)$ where

[illegible]

The multiplicity of an object is defined over an evolution rules such as:

$$| \ |_{a_i} : U \times R(U,T) \rightarrow N$$
$$(a_i, r) \rightarrow |r|_{a_i} = |m|_{a_i} = M(a_i) = u_i$$

Let C_i be the *consequent* of *the evolution rule* r_i.
Thus, the representation of the evolution rules is:

$$r_1 : a_1^{u_{11}} a_2^{u_{12}} \cdot a_n^{u_{1n}} \longrightarrow C_1$$
$$r_2 : a_1^{u_{21}} a_2^{u_{22}} \cdot a_n^{u_{2n}} \longrightarrow C_2$$
- - - ------------------------------⟶-- ------------------
$$r_m : a_1^{u_{m1}} a_2^{u_{m2}} \cdot a_n^{u_{mn}} \longrightarrow C_m$$

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

We will say that the computation has been successful if:

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

Evolution rule application phase

This phase is the one that has been implemented f ollowing different techniques. In every region withi n a p-system, the evolution rules application phase is described as follows:

Rules application to a multiset of object in a region is a transforming process of information which has input, output and conditions for making the transfor mation.

Given a region within a p-system, let U= $\{a_i | 1 \leq i \leq n\}$ be the alphabet of objects, *m* a multiset of objects over U and R(U,T) a multiset of evolution rules with antecedents in U and targets in T.

- The input in the region is the initial multiset *m*
- The output is a maximal multiset *m'*
- The transformations have been made based on the application of the evolution rules over *m* until *m'* is obtained.

Application of evolution rules in each region of P systems involves subtracting objects from the initial multiset by using rules antecedents. Rules used are chosen in a non-deterministic manner. This phase ends when no rule is applicable anymore. The trans formation only needs rules antecedents as the conse quents are part of the communication phase.

Observation.

Let $k_i \in N$ be the number of times that the rule r_i is applied. Therefore, the number of symbols a_j which have been consumed after applying the evolution rules a specific number of times will be:

$$\sum_{i=1}^{m} k_i \cdot u_{ij}$$

Definition 3.5 ***Maximal Multisets***

We say m is a maximal multiset when not anymore evolution rules can be applied to it.

Observation

Given a region R and alphabet of objects U, and R (U, T) set of evolution rules over U and targets in T.

$$r_1 : a_1^{u_{11}} a_2^{u_{12}} .. a_n^{u_{1n}} \rightarrow C_1$$

$$\ldots \qquad \rightarrow \ldots.$$

$$r_m : a_1^{u_{m1}} a_2^{u_{m2}} .. a_n^{u_{mn}} \rightarrow C_m$$

Maximal multiset is that one that complies with:

$$\bigcap_{k=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m} k_j u_{ji}\right) < u_{ki}\right] \quad [4]$$

Within a given membrane, *Let U=* $\{a_i \mid i = 1,..n\}$ be a set of objects. *Let T be* a set of targets. *Let* $\omega = a_1 a_2 .. a_n$ *be a multiset of objects and let* u_i *be the multiplicity of* a_i. *Let R(U,T) be a multiset of evolution rules with objects in U and targets in T.* Let *m* be the number of evolution rules within *R(U,T).* Let k_i be the number of times that the rule $r_i \in R(U,T)$ is applied. We define:

$$\begin{array}{lcl} \varphi : N^m & \rightarrow & N^n \\ (k_1, k_2, .., k_m) & \rightarrow & (u_1, u_2, .., u_n) \end{array}$$

"m" is the number of the evolution rules within R(U,T), "n" is the number of symbols within the given membrane.

$$\varphi(k_1, k_2, .., k_m,) \equiv \begin{pmatrix} u_{11} & u_{21} & ...u_{m1} \\ u_{12} & u_{22} & ...u_{m2} \\ \cdots & \cdots & \cdots \\ u_{1n} & u_{2n} & ...u_{mn} \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ \cdots \\ k_m \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ \cdots \\ u_n \end{pmatrix}$$

Based on definition we are interested on

$$\{k = (k_1, k_2, .., k_m) \in N^m \ / \ \varphi_1(k) = u = (u_1, u_2, .., u_n)\}$$

and $\bigcap_{k=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m} k_j u_{ji}\right) < u_{ki}\right]$ [4].

We know that there might be more than one value "k". We are just interested on one value 'k' that complies with $\varphi_1(k) = x$. This is possible as the set $K = \{k = (k_1, k_2, .., k_m) \in N^m / \varphi_1(k_1, k_2, .., k_m) = x\}$ is always included in a feasible region [4]. This feasible region is an area which contains the set *K*. There are several methods to select one element of *K* within a feasible region. These methods are studied on [4]. Because any of these methods always returns a value, we can state that this is possible. It is stated in the *algorithm for Application of Evolution Rules based on linear Diophantine equations.* [4]

IV. Building the Structure L.

Within a region, let *U*= $\{a_i \mid i = 1,..n\}$ be a *multiset of objects and let R (U, T) be* a multiset of evolution rules. *Given the set* {$k_i \in N$ *the number of times that the evolution rule* r_i *is applied over the initial multiset}, The values we are seeking are:* $k = (k_1, k_2, .., k_m) \in N^m$ For any element, $u = (u_1, u_2, .., u_n) \in N^n$, we store the value $k = (k_1, k_2, .., k_m) \in N^m \ / \ \varphi_1(k) = u$. Thus, $L[(u_1, u_2, .., u_n)] = k_1 k_2 .. k_m$

If the position $(u_1, u_2, .., u_n)$ of the structure already has information, then we do not store anything. Each entry of the structure contains the values: $k_1 .,.. k_m$. These values will indicate the number of times that an evolution rule should be applied to an initial multiset in order to obtain an extinguished multiset. The number of objects determines the coordinates of a given position in the structure; thus, as the number of objects increases so it does the dimension of the structure; i.e. working with an alphabet *V* of 2 symbols V= (a_1, a_2) means that our structure is two-dimensional. The numbers of entries existing in the structure depends on the dimension, (number of symbols of our alphabet) and the benchmark of every object. This benchmark is a value associated to every object of our alphabet. The benchmark fixes a limit for the multiplicity of the objects existing in our alphabet. In order to make the structure useful, we need to fix reasonably high benchmarks. Our structure does not store information about the values that are higher that the prefixed benchmarks.

V. The algorithm

The algorithm searches on the structure that has been previously built. The multiplicities of a given initial multiset are provided as input for the algorithm.
During compilation time the structure is build and stored in memory. During the execution time, *the algorithm just searches in the structure L the position* $(u_1, u_2, .., u_n)$

```
(1)  $u_1, u_2, .., u_n \leftarrow Multiplicity(m)$
(2)  BEGIN
(3)  $output(L(u_1, u_2, .., u_n))$
(4)  END
(5)  $REPLACE(L(u_1, u_2, .., u_n))$
```

$(u_1, u_2, .., u_n)$ *is the input value corresponding to the multiplicities of the initial multiset m.. After the output is given, another value (if it exists)*
$k' = (k'_1, k'_2, .., k'_m) \in N^m \, / \, \varphi_1(k') = u$
replaces the entry $(u_1, u_2, .., u_n)$. That way we assure that every time we execute the algorithm for the same input, we obtain a different output

VI. Conclusions

This work contributes with a new technique in finding maximal multisets from initial multisets. The technique offers a complement to be used by the traditional algorithms which calculate extinguished multisets. These algorithms find the extinguished multisets after applying a certain number of evolution rules a certain number of times. The technique presents the idea of going "backwards" from the initial multisets to the number of times that each rule should be applied. This process automatically obtains the times that each rule should be applied and no extra calculation is necessary. From there, calculating the extinguished multisets is immediate.
,In order to generalize this technique, a deeper analysis would be necessary to estimate the memory needed when certain rules are provided. This study will focus on four factors when building the structures:

- Number of evolution rules
- Applicability Benchmark of an evolution rule
- Number of symbols
- Benchmark associated to the symbols.

Within the researching area: "*membranes computing",* this work proves that not only using parallelism is beneficial in terms of performance but also memory is really helpful under certain conditions. In this case, a proper utilization of memory let us build a structure which will reside in the physical and/or the virtual memory.

VII References

[1]. Gh. Păun, "Computing with Membranes", Journal of Computer and System Sciences, 61(2000), and Turku Center of Computer Science-TUCS Report nº 208, 1998.
[2]. Gh. Păun, "Membrane computing. Basic ideas, results, applications", Pre-Proceedings of First International
Workshop on Theory and Application of P Systems, Timisoara (Romania), pp. 1-8, September , 2005.
[3]. Lingian Pan, Carlos Martin-Vide
"Solving multidimensional 0-1 knapsack problem by P systems with input and active membranesl"
Journal of Parallel and Distributed Computing
Volume 65 , Issue 12 (December 2005)
[4]. A. Arteta, L.Fernandez, J.Gil "Algorithm for Application of Evolution Rules based on linear diophantine equations" Synasc 2008
Timisoara Romania September 2008.