VIRTUAL MEMBRANE SYSTEMS

Alberto Arteta, Angel Castellanos, Nuria Gómez

Abstract: *Within the membrane computing research field, there are many papers about software simulations and a few about hardware implementations. In both cases, algorithms for implementing membrane systems in software and hardware that try to take advantages of massive parallelism are implemented. P-systems are parallel and non deterministic systems which simulate membranes behavior when processing information.*

This paper presents software techniques based on the proper utilization of virtual memory of a computer.

There is a study of how much virtual memory is necessary to host a membrane model.

This method improves performance in terms of time.

Keywords: P-systems, Parallel systems, Natural Computing, evolution rules application, set of patterns, Virtual structure.

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

Introduction

Membrane computing is a parallel and distributed computational model based on the membrane structure of living cells [Pãun 2000]. This model has become, during these last years, a powerful framework for developing new ideas in theoretical computation. The main idea was settled in the base of connecting the Biology with Computer Science in order to develop new computational paradigms. P-systems are the structures that reproduce the information processing occurring in the living cells. Nowadays, it can be found that several P Systems simulators are much elaborated [Pãun 2005]. At this point, there is a problem with the parallelism synthesized in [Pãun 2005] by: "parallelism a dream of computer science, a common sense in biology". This is the reason why, Pãun avoids "to plainly saying that we have 'implementations of P systems", because of the inherent non-determinism and the massive 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 [Pan, Martin 2005].

Also, Membrane computing is currently used to model other parallel and distributed systems as Grid [Qi, Li, Fu, Shi, You 2006].

An overview of membrane computing software can be found in [Ciobanu, Pérez-Jiménez, Ciobanu, Pãun 2006], or tentative for hardware implementations [Fernández, Martínez, Arroyo, Mingo 2005], or even in local networks [Ciobanu, Wenyuan 2004].

Work structure

This paper is structured as follows:

We will define some concepts as: P-systems, Extinguished multisets, multisets of objects, multiplicity of an object and evolution rules.

Reviewing the concept of patterns [Arteta, Castellanos, Martinez 2010].

Creation of a main n-dimensional structure in a virtual auxiliary n-dimensional structure. That main structure is 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 an extinguished multiset.

Definitions

Here are some helpful definitions to understand this work. More info about these definitions can be found in [Arteta, Castellanos, Martinez 2010]

Definition Transition P Systems

A Transition P System of degree *n*, *n* > 1 is a construct

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

Where:

V is an alphabet; its elements are called objects;

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*;

$\omega_i \; 1 \le i \le n$, 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 $\prod$

Definition 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:

$M : U \to N$

$a_i \to u_1$

Where a_i is an object and u_i its multiplicity.

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

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

Definition Evolution rule with objects in *U* and targets in *T*

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*'

Definition The set of evolution rules

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

Definition 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:

$$\left|\ \right|_{a_i} : U \times M(U) \to N$$
$$(a_i, m) \to \left|m\right|_{a_i} = n \mid (a_i, n) \in m$$

Definition 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 $m \in M(U), c \in M(UxT)$ *and* $\delta \in \{$*to dissolve, not to dissolve*$\}$

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

$$\left|\ \right|_{a_i} : U \times R(U,T) \to N$$
$$(a_i, r) \to \left|m\right|_{a_i} = n \mid (a_i, n) \in m$$

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

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

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:

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

Definition Extinguished Multisets

Given a region *R*, let *U* be an alphabet of objects $U = \{a_i \mid 0 < i \leq n\}$.

Let $M(U) = \{(a_i, u_i) \mid a_i \in U\ u_i \in \mathrm{N}\ 0 < i \leq n\}$ the set of all the multisets over *U*. Let $x \in M(U)$ be a multiset of objects over *U* within *R*. Let $R(U,T) = \{r_i\ i \in \mathrm{N} \mid \exists m \in \mathrm{N}\ i \leq m\}$ be a set of evolution rules. $r_j = (x_j, c_j, \delta) \in R(U,T)$ *and* $x_j = \{(a_{ji}, u_{ji})\ \ i, j \in \mathrm{N} \mid \exists n, m \in \mathrm{N}\ i \leq n,\ j \leq m\}$

Let $k_j \in \mathrm{N}$ the number of times that the rule $r_j \in R(U,T)$ is applied over *x*. We say *x* is an Extinguished multiset if and only if :

$$\bigcap_{j=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m}(k_j \cdot u_{ji}) \leq u_{ji}\right)\right]$$

Observation

In other words, *m* is an Extinguished multiset if and only if is not possible to apply any more evolution rules over it.

Patterns created from an evolution rules multiset.

In this section we show the patter definition. More information about patters can be found in [Arteta, Castellanos, Martinez 2010]

Definition Pattern

$N \otimes N \rightarrow A \quad A \subseteq \mathrm{P}(\mathrm{N})$

$[i,j] = \{k \in N \mid i \leq k \leq j\ \ i, j \in N\}$,

Definition Additive translation of a pattern

$n+[i,j]=[n+i,n+j]$

Definition Subtractive translation of a pattern

$n-[i,j]=[n-i,n-j]$

$n - i = 0 \quad n \leq i$

Definition Set of pattern $S \subseteq \mathrm{P}(\mathrm{N})$ is a Set of patterns is defined as the set:

$$\left\{[a_i, b_j] \mid \exists n, m \in \mathrm{N}, i \leq n\ j \leq m a_i, b_j, i, j \in \mathrm{N}\right\}$$

Definition Order in a set of patterns.

Let $S = [[a_1, a_2], .. [a_3, a_4], .. [a_{n-1}, a_n]]$

We define:

$$f_{number} : \mathrm{N} \otimes \mathrm{P}(\mathrm{N}) \rightarrow \mathrm{P}(\mathrm{N})$$
$$(i, S) \quad \rightarrow [a_i, a_{i+1}] \quad S \in \mathrm{P}(\mathrm{N}), i \in \mathrm{N}$$

$$\Rightarrow f_{number}(i, S) = [a_i, a_{i+1}]$$

Note: From now on, we will denote $f_{number}(i, S)$ *as* $S[i]$

Definition Inclusion in a set of patterns

Let S be a set of n patterns; and let $x=[x_1,x_2,...x_n]\in \mathbb{N}^n \quad x\in S \Leftrightarrow x_i \in S[i]\ \forall i\in \mathbb{N} \quad i\le n$

Definition Set of set of patterns

A Set of set of patterns $SS=\{S_i \quad i\in \mathbb{N}\ S_i \text{ is a set of patterns} \mid \exists n\in \mathbb{N}\ i\le n\}$.

Observation

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

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

$r_2 : a_1^{u_{21}} a_2^{u_{22}} . a_n^{u_{2n}} \to C_2$

... $\to$....

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

There is a Set of set of patterns $SS_{R(UT)}$ associated to it. This set of set of patterns contains all the possible extinguished multisets and it is obtained by expanding the formula included in the definition of extinguished multiset:

$$\bigcap_{l=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m}(k_j \cdot u_{ji}) \le u_{li}\right)\right]$$ [Arroyo, Luengo 2003]

$$\left(\begin{array}{l}
[[0,u_{11}],[0,u_{12}]...[0,u_{1n}]],[[0,u_{11}],[0,u_{12}]...[0,u_{2n}]],...[[0,u_{11}],[0,u_{12}]...[0,u_{mn}]],\\
...\\
[[0,u_{11}],[0,u_{22}]...[0,u_{1n}]],[[0,u_{11}],[0,u_{22}]...[0,u_{2n}]],...[[0,u_{11}],[0,u_{22}]...[0,u_{mn}]],\\
...\\
[[0,u_{11}],[0,u_{m2}]...[0,u_{1n}]],[[0,u_{11}],[0,u_{m2}]...[0,u_{2n}]],...[[0,u_{11}],[0,u_{n2}]...[0,u_{mn}]],\\
...\\
[[0,u_{21}],[0,u_{12}]...[0,u_{1n}]],[[0,u_{21}],[0,u_{12}]...[0,u_{2n}]],...[[0,u_{21}],[0,u_{12}]...[0,u_{mn}]],\\
...\\
[[0,u_{21}],[0,u_{22}]...[0,u_{1n}]],[[0,u_{21}],[0,u_{22}]...[0,u_{2n}]],...[[0,u_{21}],[0,u_{22}]...[0,u_{mn}]],\\
...\\
[[0,u_{21}],[0,u_{m2}]...[0,u_{1n}]],[[0,u_{21}],[0,u_{m2}]...[0,u_{2n}]],...[[0,u_{21}],[0,u_{n2}]...[0,u_{mn}]],\\
...\\
[[0,u_{m1}],[0,u_{12}]...[0,u_{1n}]],[[0,u_{1n}],[0,u_{12}]...[0,u_{2n}]],...[[0,u_{m1}],[0,u_{21}]...[0,u_{mn}]],\\
...\\
[[0,u_{m1}],[0,u_{22}]...[0,u_{1n}]],[[0,u_{1n}],[0,u_{22}]...[0,u_{2n}]],...[[0,u_{m1}],[0,u_{22}]...[0,u_{mn}]],\\
...\\
[[0,u_{m1}],[0,u_{m2}]...[0,u_{1n}]],[[0,u_{1n}],[0,u_{m2}]...[0,u_{2n}]],...[[0,u_{m1}],[0,u_{2n}]...[0,u_{mn}]],
\end{array}\right)$$

Observation:

The cardinal of $SS_{R(U,T)}$ $|SS_{R(U,T)}|=n^m$ where $n=|U|$ and $m=|R(U,T)|$

is

Definition Translation of a Set of set of patterns

$$N^n \otimes N^{n^m} \to N^{n^m}$$

$$(x_1, x_2, .., x_n) - \begin{pmatrix} [[0,u_{11}],...[0,u_{1n}]],...[[0,u_{11}],..[0,u_{mn}]], \\ ... \\ [[0,u_{m1}],..[0,u_{mn}]],...[[0,u_{m1}],..[0,u_{mn}]], \end{pmatrix} = \begin{pmatrix} [[x_1 - u_{11}, x_1],...[x_n - u_{1n}, x_n]],...[[x_1 - u_{11}, x_1],..[x_n - A_{mn}, x_n]], \\ ... \\ [[x_1 - u_{m1}, x_1],..[x_n - u_{mn}, x_n]],...[[x_1 - u_{m1}, x_1],..[x_n - u_{mn}, x_n]], \end{pmatrix}$$

Evolution rules linear functions

In [Arteta, Castellanos, Martinez 2010] several functions are defined to build a structure in the physical memory of a computer. Our work defines these virtual functions to build the structure in virtual memory. During this part we will define the necessary function that will allow us to create the virtual structure. The structure is placed on virtual memory. The function will be referred to as "virtual" evolution rules linear isomorphism. These functions will contain itself other function as follows.

Virtual linear evolution rules function

Based on the previous definitions, it is possible to define a function which will be the key for building a linear structure and then to allocating it in virtual memory. This function will be the composition of two different functions.

Virtual linear Multisets function

The function is defined as follows:

$$\Phi_{1virt}: N^n \to P(N^m)$$

$$[x_1, x_2, .., x_n] \to \begin{bmatrix} k_1, k_2, .. k_m \\ k_1', k_2', .. k_m' \\ ... \\ k_1'^p, k_2'^p, .. k_m'^p \end{bmatrix}$$

Given an input $x = (x_1, x_2, .. x_n) \in N^n$, it returns the set of numbers $k = (k_1, k_2, .. k_m) \in N^m / \varphi_1(k) = x$

Virtual linear Pattern function

$$\Phi_{2virt}: P(N^m) \to N^n$$

$$\begin{pmatrix} [[0,u_{11}],...[0,u_{m1}]],..,[[0,u_{11}],..[0,u_{mn}]], \\ ... \\ [[0,u_{12}],..[0,u_{m1}]],..,[[0,u_{12}],..[0,u_{mn}]], \end{pmatrix} \to \begin{bmatrix} x_1, x_2, .., x_n \\ x_1', x_2', .., x_n' \\ ... \\ x_1'^p, x_2'^p, .., x_n'^p \end{bmatrix}$$

Given a set of set of patterns as the input it returns a set of numbers $x=(x_1,x_2,..x_n)\in\mathbb{N}^n$. The elements of this resulting set are all the combinations of all the possible

$x^j=(x_1,x_2,..x_n)\in\mathbb{N}^n$ where $x_i^j\in pattern(i)\in SP(j)$ of a set of patterns contained in the matrix of set of patterns.

Building virtual linear structures from the multisets isomorphism.

Creation of the virtual linear structure $L_{\Phi virt}$

Given $V=\{\{X_i \mid i=1,..n\}$ be a multiset of symbols and given $R(U,T)$, 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}. We build the evolution rules function. $\Phi_{virt}=(\Phi_{1virt}\circ\Phi_{2virt})(P_{R(U,T)}(A_{ij})$

Once the virtual function has been constructed, the new virtual linear structure must be built this way:

$$L_{virt}\left[\Phi_{2virt}\left(P_{R(U,T)}(A_{ij})\right)\right]=\Phi_{1virt}\circ\Phi_{2virt}\left(P_{R(U,T)}(A_{ij})\right)append[L_{virt}\left[\Phi_{2virt}(M)\right]]$$

When $L_{\Phi virt}$ already has a value, then the new values are included and are appended to the existing ones. This means that in each entry of the virtual linear structure, there will be different values $(k_1,k_2,..,k_m)append[(k_1^{'},k_2^{'},..,k_m^{'})]append...append[(k_1^{'p},k_2^{'p},..,k_m^{'p})]$.

A concatenation of $(k_1,k_2,..k_m)$values will be included in each cell of the virtual linear structure. This linear structure has to comply with having all possible numbers up to a combination of benchmarks reasonably high. Each symbol $\{X_i \mid i=1,..n\}$ will have a benchmark. The combination of all the benchmarks will define the number of entries that the linear structure has. Each entry will store a concatenation of 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. After proving the consistency of the physical structure [Arteta, Castellanos, Martinez 2010] proving the consistency of the virtual one is trivial.

Now there is enough information to build the physical linear structure L_Φ in this example. A way to build the structure could be:

At this moment we are able to build the virtual linear structure. Each cell a concatenation of values will be stored instead of just one value.

The structure is built as follows:

$L_{\Phi virt}(x)=$

$L_{\Phi virt}(0,1)=0,0,1\ L_{\Phi virt}(1,0)=0,0,1\ L_{\Phi virt}(1,1)=0,0,1\ L_{\Phi virt}(2,0)=0,0,2$

$L_{\Phi virt}(2,1)=0,1,0\ L_{\Phi virt}(2,2)=0,1,0\quad 0,0,2\ L_{\Phi virt}(3,2)=0,1,0\ L_{\Phi virt}(3,3)=0,1,0$ In this way, the

$L_{\Phi virt}(4,3)=1,0,0\quad 0,1,1 L_{\Phi virt}(5,4)=1,0,1\quad 0,1,2\quad 0,2,0....$

structure will be created and allocated in the virtual memory.

Algorithm

(1) $X,Y \leftarrow Multiplici\ ty(R(U,T))$

(2) *BEGIN*

(3) *output* $(L_{\Phi}(X,Y))$

(4) *END*

(5) $(L_{\Phi}(X,Y)) \leftarrow random\ (L_{\Phi virt}(X,Y))$

The algorithm search in the virtual structure the position(*X,Y*) which are the input values corresponding to the multiplicities of the initial multiset. When the value is returned, a new value coming from the virtual structure overwrite the value stored in the position (*X,Y*).

Conclusion

This paper is a continuation of [Arteta, Castellanos, Martinez 2010]. Here, the author analyzes the resources to use to build a structure in the RAM of the computer. This paper describes a method to do so in a virtual memory device. When there are limitations on physical memory, a proper use of the virtual one is recommended. The way to build the structure determines the proper performance of the method.

Bibliography

[Arroyo, Luengo 2003] F. Arroyo, C. Luengo, A.V. Baranda and L.F. Mingo. A software simulation of transition P systems in Haskell. International Workshop Membrane Computing, Curtea de Arges (Romania), August 2002, Springer-Verlag, Vol 2597, pp. 19-32, Berlin, 2003.

[Arteta, Castellanos, Martinez 2010] A. Arteta, A, Castellanos, A. Martinez. Membrane computing: Non deterministic technique to calculate extinguished multisets. International Journal "Information Technologies and Knowledge", Vol. 4, Number 1, pp. 30-40. 2010.

[Arteta, Fernandez, Gil 2008] A. Arteta, L.Fernandez, J.Gil. Algorithm for Application of Evolution Rules based on linear diophantine equations. Synasc 2008. Timisoara Romania September 2008.

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

[Ciobanu, Wenyuan 2004] G. Ciobanu, G. Wenyuan. A P system running on a cluster of computers. Proceedings of Membrane Computing. International Workshop, Tarragona (Spain). Lecture Notes in Computer Science, Vol 2933, Springer Verlag, pp. 123-150, Berlin, 2004.

[Fernández, Martínez, Arroyo, Mingo 2005] L. Fernández, V.J. Martínez, F. Arroyo, L.F. Mingo. A Hardware Circuit for Selecting Active Rules in Transition P Systems. Proceedings of International Workshop on Theory and Applications of P Systems.Timisoara (Romania), September, 2005.

[Pan, Martin 2005] L. Pan, C. Martin-Vi de. 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)

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

[Păun 2005] 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.

[Qi, Li, Fu, Shi, You 2006] Zhengwei Qi, Minglu Li, Cheng Fu, Dongyu Shi, Jinyuan You. Membrane calculus: A formal . method for grid transactions. Concurrency and Computation: Practice and Experience Volume18,Issue14 , Pages1799-1809. Copyright © 2006 John Wiley & Sons, Ltd.

Authors' Information

Alberto Arteta –Associate ProfessorTechnical University of Madrid. aarteta@eui.upm.es

Angel Castellanos –Departamento de Ciencias Basicas aplicadas a la Ingeniería Forestal. Escuela de Ingeniería Técnica Forestal. Technical University of Madrid, Avda. de Ramiro de Maeztu s/n 28040 Madrid, Spain. angel.castellanos@upm.es

Nuria Gómez Blas–Departamento de Organización y Estructura de la Información. Escuela Universitaria de Informática. Technical University of Madrid, Crta DE Valencia KM. 7, 28031 Madrid, Spain. ngomez@eui.upm.es