Parallel Algorithm for P Systems implementation in Multiprocessors

F. Javier Gil, Luis Fernández, Fernando Arroyo, J. Alberto de Frutos

Natural Computing Group, Universidad Politécnica de Madrid, Spain
(Tel: 34-91-336-5087; Fax: 34-91-336-7520)
{jgil, setillo, farroyo, jafrutos}@eui.upm.es

Abstract: The Transition P systems perform a computation through transition or evolution step between two consecutive configurations. Each evolution step is made up of two phases: on the first phase the evolution rules are applied, and a second phase of communication between membranes. At the time of designing architectures hardware and/or software for the implementation of Transition P systems, has been detected the necessity of having algorithms that can know beforehand the required time to apply a set of active evolution rules on a multiset inside a membrane. This is very important as it allows to determinate the balanced number of membranes to be located in each processor in the implementation of distributed architectures of P systems in order to achieve optimal evolution step time with minimal resources.

In this work we introduce a parallel algorithm for application of active rules in a membrane oriented towards the implementation of Transition P systems in multiprocessors hardware architectures. Given a set of active rules R, the algorithm is based on power set of R (the set formed by all the subsets of R) and on the rules elimination algorithm (applying them to its maximum applicability benchmark). This new algorithm complies the requisites of being nondeterministic, parallel, and what is very important, the algorithm is time delimited because it is only dependant on the number of evolution rules placed in a membrane.

Keywords: Natural computing, Membrane computing, Evolution Rules Application Algorithms

I. INTRODUCTION

Membrane computing are a distributed highly parallel computational model introduced by G. Păun [1] inspired on basic features of biological membranes and the observation of biochemical processes. In this model, membrane contains multisets of objects –usually strings of symbols- which evolve according to a given set of evolution rules. Applying these evolution rules in a nondeterministic maximally parallel way the system changes from a configuration to another one making a computation. This computing model has become, during last years, an influential framework for developing new ideas and investigations in theoretical computation.

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" [2]. 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.

Frequently, P systems are implemented and simulated on a standard computer using an existing or a custom simulator (such as the Membrane Simulator [3], or one of the proposed distributed software simulators [4]. Ciobanu [5] says: "As there do not exist, up to now, implementations in laboratories (neither in vitro or in vivo nor in any electronically medium), it seems natural to look for software tools that can be used as assistants that are able to simulate computations of P systems". Additionally in [2] Păun says: "An overview of membrane computing software can be found in literature, or tentative for hardware implementations, or even in local networks is enough to understand how difficult is to implement membrane systems on digital devices"

P systems were usually simulated on a single processor machine and their inherent parallelism could therefore not be fully exploited. Păun says that: "we avoid to plainly say 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 -but which can be implemented on a dedicated, reconfigurable, hardware [...] or on a local network". Thereby, there exists many P systems simulators in bibliography but "the next generation of simulators may be oriented to solve (at least partially) the problems of storage of information and massive parallelism by using parallel language programming or by using multiprocessor computers" [5].

Related work

At this moment several algorithms for application rules in P systems exist [3, 6 and 7] however, they are sequential algorithms. Tejedor in [7] introduce an evolution rules application algorithm on a multiset of objects to use in the P system implementation for digital devices based on the elimination of active rules. The algorithm reaches a certain degree of parallelism, since one rule can be simultaneously applied several times in a single step, but it is a kind of sequential algorithm. This solution is, of course, a minimal parallelism solution, but it is interesting because it is the first algorithm whose time is only limited by the number of rules, not by the objects multiset cardinality.

They have been proposed parallel solutions also [8 and 9], but these algorithms get poor performance due to the high number of collisions between request of the rules.

Tejedor [10] proposes a software architecture where several membranes are located in each processor, proxies are used to communicate with membranes located in different processors. In addition, it establishes a set of equations that they allow to determine 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. Consequently, 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 can be executed in a delimited time are required, independently of the object multiset cardinality inside the membranes.

This work presents an algorithm free of collisions, time delimited and massively parallel for application of active rules inside a membrane in transition P systems. After this introduction, appears the active rules application algorithm description, including the synchronization between processes and the efficiency analysis. Finally, the conclusions are included.

II. DESCRIPTION OF THE ACTIVE RULES APPLICATION ALGORITHM

This section describes a time delimited parallel algorithm for application of active rules in a membrane oriented towards the implementation of Transition P systems in multiprocessors hardware architectures. The initial input of the algorithm is the set of active evolution rules (denoted by R) for the corresponding membrane -the rules are applicable and useful- and the initial membrane multiset of objects. The final results are the multiset of evolution rules applied and the multiset of objects obtained by the rules application.

As it is known, the powerset of the active rules -denoted by P(R)- is the set whose elements are all the subsets of R. In our case, we are interested in all elements, except the empty set.

As it is detailed ahead, we propose one process for each member of the powerset of R (rule process) and one more controller process that simulate the membrane containing the multiset of objects (membrane process).

The algorithm idea is that each member of the powerset of R -except the last one rule- randomly proposes in an independent manner, a multiset to be consumed from the membrane multiset. Then, in parallel, there is a choice in binary tree and the "winner" proposed multiset is subtracted from the membrane multiset. This step is executed many times as the number of active rules minus one. This provides to all rules the possibility of being applied.

After this process, the last active rule determines and applies its maximal applicability benchmark over the membrane multiset, subtracting the correspondent multiset from the membrane multiset. At this point, rules that are not applicable over the new membrane multiset finish their process execution -obviously, including the last active rule and all the subsets contain it-. The remaining active rules come back to the starting point, and propose a new multiset to be consumed again. This process is repeated until it is not left any rule applicable over the membrane multiset.

Next we explain more in detail each one of these phases:

Phase 1 Membrane initialization. The membrane generates P(R) and initializes the data structures.
This phase is performed only by the controller process, while rules are waiting to second phase (synchronization point A).

Phase 2 Evolution rules initialization. Each element of P(R) -except the last active rule- determines its maximal applicability benchmark over the membrane multiset.
This phase is performed in parallel by every member of P(R) except the last active rule. The controller process -membrane- waits until next phase (synchronization point B).

Phase 3 Initialization of iterations counter. This counter is used to provide all rules the possibility of being applied before applying the last rule to its maximal applicability benchmark.
This phase is only performed by the membrane.

Phase 4 Multiset proposition. Each active element of P(R) proposes in a randomly manner one multiset of objects to be consumed from the membrane multiset. The proposed multiset can be the empty multiset or the scalar product of the rule multiset antecedent by a natural number chosen in a random manner between 1 and its maximal applicability benchmark. Each applicable member must wait until their neighbor finish (synchronization point C).
This phase is performed in parallel for the elements of powerset of R, except the last active rule.

Phase 5 Selection of the applicable component. The selection of the applicable member of P(R) is made in parallel two by two in a random way.
This stage is performed in parallel for every active element of the powerset of R. Membrane must wait (synchronization point D) until the selection finishes.

Phase 6 Selection application. Membrane process applies the member of P(R) chosen in the previous phase, subtracting the correspondent multiset from the membrane multiset. In addition, it increases the counter of iterations: if it has reached the number of active rules minus one, then the last active rule is applied over the membrane multiset to its maximal applicability benchmark. Else a new iteration is made, running back stage 4. Moreover, the membrane process indicates to the rule processes the executed operation and finally, it initializes information about its active evolution rules for the next loop in the algorithm.
This phase is performed only by membrane process while other processes wait. Rule processes must wait to start checking their halting condition (synchronization point E).

Phase 7 Checking rules halt. Each one of the elements of P(R) accumulates the number of applications made over the membrane multiset. Moreover, it computes its maximal applicability benchmark over the new resting membrane multiset for the next iteration and, if it is bigger than 0, they pass to the state in which they can propose and indicate it into the active evolution rules data structure. Otherwise, they finish their execution.
This phase is performed in parallel by every evolution rule except into the access to the active evolution rules data structure (synchronization point F). Membrane is waiting until phase 8 (synchronization point G).

Phase 8 Checking membrane halt. Membrane checks if there exists some active rule for the next loop and, in this case, it returns to propose multisets by the elements of P(R). If so, it comes back to stage 3 for a new iteration, otherwise it finishes the execution.
This phase is performed only by the membrane and the rules processes wait (synchronization point H) for coming back to phase 4 -if they are active for next loop- of for finish their execution.

Next we will deal with two different aspects for the exposed general idea: the phase's synchronization and finally efficiency analysis.

Synchronization Design

Accordingly whit the previous explanation, these phases shared between the two different processes types, evolution rules and membrane, as it can be observed in tables 1 and 2.

```
(1): Membrane initialization
REPEAT
   (3): Counter initialization
   REPEAT
     (6): Selection application
      & last active rule application
   UNTIL counter = |R| - 1
   (8): Checking membrane halt
UNTIL End
```

Table 1: Process type membrane

```
(2): Member of P(R) initialization
REPEAT
   (4): Multiset proposition
   (5): Selection of P(R) element
   (7): Checking rules halt
UNTIL End
```

Table 2: Process type rule

Both processes types are not disjoint and they must preserve the following synchronizations (Fig. 1 presents the activity diagram showing the needed synchronization in the different phases for the process membrane and two evolution rules processes).

Efficiency Analysis

Observing the process type membrane, it can be observed that this algorithm is executed at the most so many times as rules exist initially (the cardinality of the set of rules R) since, in each iteration, at least one rule is eliminated in the worse case. In each iteration, they are made (|R| - 1) internal iterations, and one element of P(R) is selected and possibly applied.

In each one of these internal iterations are executed simple instructions and the selection in binary tree of an applicable element. Therefore the order is $\log_2(|P(R)|)$, that it is equivalent to |R|.

Accordingly, in the worst case (assuming that the chosen item never be executed), the number of steps made will be:

$$(R-1)^2 + (R-2)^2 + \dots + 1 \qquad (1)$$

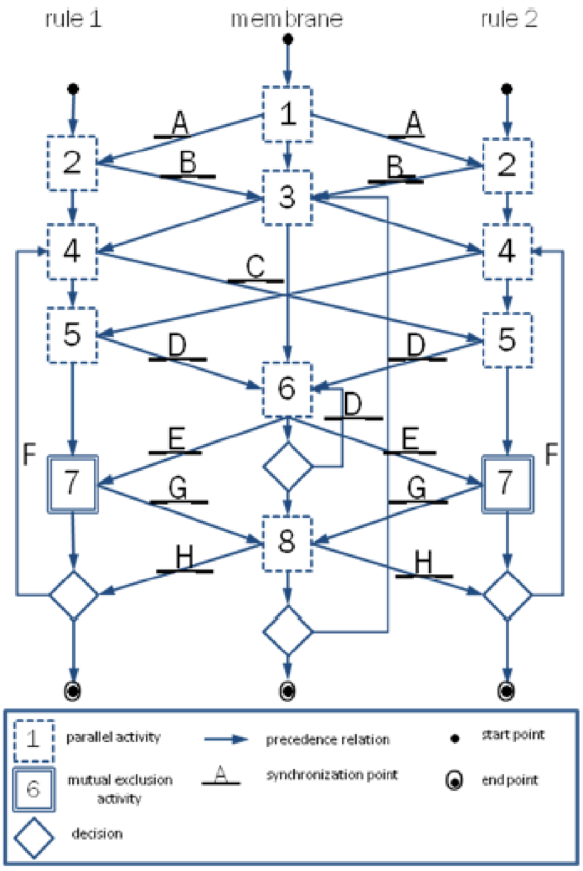

Fig.1. Activity diagram for the membrane and two evolution rules

Developing the expression (1) is concluded that the number of iterations in the worst possible case (all members of P(R) proposed the empty multiset) is the order of $R^3/3$.

Consequently, we can conclude that the complexity order of the proposed algorithm -in the worse case- is $R^3/3$, but better results can be expected experimentally than the ones obtained theoretically, because of two factors: 1) the probability of proposing a empty multiset is minimal, and 2) is possible that in a same loop iteration more than one rule becomes inactive.

III. CONCLUSION

This paper introduces an algorithm of active rules application based on powerset of R and the elimination of rules. The most important characteristics of this algorithm are: to be massively parallel, free of collisions and its execution time is delimited, because it only depends on the number of rules of the membrane.

Because of these characteristics, we think that the presented algorithm can represent an important contribution for the problem of the application of rules in membranes, because it presents high productivity and allows estimate the necessary time to execute an evolution step. Additionally, the last one allows make important decisions related to the implementation of P systems in microprocessors, as those related to architecture.

REFERENCES

[1] Păun, G. (1998), Computing with Membranes. Journal of Computer and System Sciences, 61 (2000), and Turku Center of Computer Science-TUCS Research Report, No 208
[2] Păun, G. (2005), "Membrane computing. Basic ideas, results, applications" Pre-Proceedings of First International Workshop on Theory and Application of P Systems, Timisoara (Romania): 1-8
[3] Ciobanu, G. and Paraschiv, D. (2002), Membrane software. A P system simulator. *Fundamenta Informaticae*, 49 (1-3): 61-66
[4] Ciobanu, G. and Wenyuan, G. (2003), A parallel implementation of the transition P systems. Proceedings of the MolCoNet Workshop on membrane Computing, vol. 28/03: 169-169
[5] Ciobanu, G. Pérez-Jiménez, M. Păun, G. (2006) Applications of Membrane Computing. Natural Computing Series, Springer Verlag, October 06
[6] 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)
[7] Tejedor, J. L. Fernández, F. Arroyo *et al* (2007) "Algorithm of Active Rules Elimination for Evolution Rules Application". 8th WSEAS, Vancouver (Canada)
[8] Fernández, L. Arroyo, F. Tejedor, J. *et al* (2006) "Massively Parallel Algorithm for Evolution Rules Application in Transition P System". WMC7, Leiden (The Netherlands).
[9] Gil, F. J. Fernández, L. Arroyo, F. *et al* (2007) "Delimited Massively Parallel Algorithm based on Rules Elimination for Application of Active Rules in Transition P Systems" i.TECH-2007. Varna (Bulgaria).
[10] Tejedor, J. L. Fernández, F. Arroyo et al (2007) "An Architecture for Attacking the Bottleneck Communications in P systems". AROB'07. Beppu (Japan).