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

COMMUNICATION LATENESS IN SOFTWARE MEMBRANES

Miguel Angel Peña, Jorge Tejedor, Juan B. Castellanos, Ginés Bravo

***Abstract**: This paper presents a study about the time of the communication phase of the P System implemented as software. After presenting the membranes as software components, is studied how communicate with each other, running on the same processor or another. We study the times and delays that occur if communication takes place via network. It presents the theoretical formulas governing the study, empirical studies, and adjustment factors which depend on communications. Finally, some examples that show facts for some P System; the distribution of membranes on several computers improves global times.*

***Keywords**: P System, Membrane Systems, and Natural Computation.*

***ACM Classification Keywords**: F.1.2 Modes of Computation, I.6.1 Simulation Theory, H.1.1 Systems and Information Theory*

Introduction

In 1998 Gheorghe Paun introduced membrane computing [Paun, 2000] as a parallel computing model based on the biological notion of the cell. On the original model, there have been several variations in order to solve various problems, and improve computation times, in order to solve complex problems such as NP-complete, in times similar to the polynomial.

The idea behind a membrane system is based on the permeability of the same, and the internal changes taking place. Depending on the elements that are working, we can distinguish two main types, those which manipulate objects, and working with strings. The behavior is similar in both cases. In parallel, each membrane performs a series of rules with objects or strings you have, resulting in other ob-jects or strings, which, using the permeability of the membrane can move to other membrane if they indicate their transformation rules. The great advantage of these systems is that each membrane is run independently of the others, so the runtime does not depend on the number of membranes.

From this model, many researchers have developed software implementations, with different points of view. Some of them have been based on hardware architectures, such as [Petreska, 2003], [Fernández, 2005] or [Martínez, 2006]. Others were based on software simulation, in different languages, such as [Cordón-Franco, 2004] or [Malita, 2000] in Prolog, [Suzuki, 2000] LISP based, [Arroyo, 2003] Haskell based, [Balbontín-Noval, 2002] Scheme based, [Nepomuceno-Chamorro, 2004] Java based. These simulators use only one processor to perform operations. For more performance software implementations should use the idea of a distributed architecture, as proposed [Ciobanu, 2004] in C++ and [Syropoulos, 2003] in Java. What is not mentioned in any of these implementations is how to make the distribution of membranes per processor. To carry out this distribution is necessary to know the time it takes communication to make the best possible distribution.

P- System definition

The first definition of a P System was made for Paun [Paun, 2000], a defined a Transition P System:

Definition 1. A Transition P System is $\prod = (V, \mu, \omega_1, \ldots, \omega n; (R_1, \rho_1), \ldots, (R_n; \rho_n); i_0)$, where:

V is an alphabet (composed of objects),

μ is the membrane structure with n membranes

ω_i are the multiset of symbols for the membrane i.

R_i are the evolution rules for the membrane i.

P_i are the priority of rules for the membrane i.

i_0 indicates a membrane, which is the membrane of system output.

Running P-System is made through configurations. The following phases are making on each configuration of each membrane in parallel and no deterministic way:

Determining the set of utility rules: On this micro-step are evaluating all evolution rules of the membrane in order to determine which are useful. A rule is useful when all membranes exists in the P-System that are indicated in its consecuent

Determining the set of applicable rules: It will be necessary evaluate all the evolution rules of the membrane for identify those that meet that its predecessor is contained in the multiset of objects of the region.

Determining the set of actives rules: Intersection of two previous sets conform the entrance group to this micro-step. Each one of rules of the set must be processed, to determine which meets the condition of active rule. To determine if a rule meets that condition, is necessary check there is not another rule with higher priority that belongs to the useful and applicable rules set.

Non deterministic distribution of objects of the region between its active rules and application: In this micro-step, copies of present objects in the multiset of the region are distributed between active evolution rules of the same. Copies of objects that are assigned to each rule, match with those of the multiset that results from scalar product of a number between minimum and maximum bound of applicability of those rule and its predecessor. This distribution process is made on a non-deterministic way. Moreover, at the end of it, objects no assigned to any rule forms a multiset and they will be characterized because they do not contained to any predecessor of rules. The result of the distribution is a multiset of actives rules, where multiplicity of each rule defines the times that would be applied, and therefore, indirectly through its predecessors, objects are assigned to the multiset of the region. Objects used are eliminated and generate new objects that are destined for a membrane.

Transfering of multiset of generated objects or communication phase: In this micro-step, the new objects generated on the previous micro-step whose indicator of destiny membranes was 'in' or 'out', must be transferred to its corresponding membranes. Each membrane, will have unused objects in its application, together with those that result fo the applying of rules and that have this membrane as destiny.

Implementation software

At [Gomez, 2007] is defined a framework for Transition P-System. Following this model has been done a software implementation on java. Each computer, with one processor, will contain a number of membranes that need not be equal. Furthermore, each computer will be known where all the membranes are. Computers will be connected via Ethernet. On each processor, will be running in a secuencial way each one of the phases that were indicated previously, on each one of membranes, in a secuencial way too. At finished each phase it will make synchronization between processors with the purpose that all of them will begin the next phase simultaneously. We used some computer at 0,7 GHz, with 0,5 MB of RAM. They were connecting with a 100 Ethernet.

Communication Phase

With the application of rules, is possible that one membrane generates objects that are destinated to another membrane taking advantage the characteristic of permeability of the membrane. Because that, the new generated object must be sended, on communication phase, to its destiny membrane. On implemention over distributed software, is possible that this destiny membrane is in the same or in another machine. Because that, the sending method will be different for each one of two cases that we find. In the case of sending over the same processor, and therefore the share memory, will take place inmediatly, with the processor in charge of the whole process. In the case of source and target membranes, are on distinct processors, is needed send objects through the network.

Because there are two types of connections between membranes – internal and external to the processor-, times of the communication phase not only depend on the number of objects to interchange, but also of disposition of membranes on the processors.

To analize the communication between processors, firstly, we should define the message to send through the net. Although there may be multiple formats to the communication message, we go to use a simplification of it:

- The first four bytes represent the destiny membrane of objects. It reserves the highest number of membranes, to indicate that the message is used for a different use, like to synchronize executing, dissolution of membranes,.. The use of four bytes limits the number to 2^{24} membranes.
- The next byte represents the object of whom will exist n repetitions to add. Therefore, it exists a limitation to a vocabulary of 2^8 objects.
- The next two bytes, represent the repetitions that are being transfered, of previously indicated object. Therefore, it can transfer, only 2^{16} equal objects.
- Symbol and repetitions block can appear many times as it likes, until find the special object corresponding to value 28.

Hence, message size depends on quantity of distinct objects that must be sending, but like minimum will be 8 bytes. It has been made many tests to comparate times it takes to send a determinate number of bytes. For taking time, it has been synchronized watchs on both computers, on a level to seconds, and it has been made 100000 repetitions in a row of each test, in order that synchronism error was negligible. Results are show on figure 1 – left. On it observes that distinct teste on the same conditions, presents distinct results. It is because when executions are made over software, several factors influence the time:

- The processor executes several tasks and depending of Operative System and task manager, it can delay execution of process of execution of P-System.
- It can perform other input-output operations
- May be necessary to release memory and the interchange from swap area.
- There may be network congestion, as indicated by Ciobanu [Ciobanu, 2004].
- The Java Virtual Machine can introduce similar delays too

However, removing data that clearly show excessive delays, it can obtain values to adjust (Figure 1 right):

$$\text{Time} \approx 0{,}0023 \text{ Bytes}$$

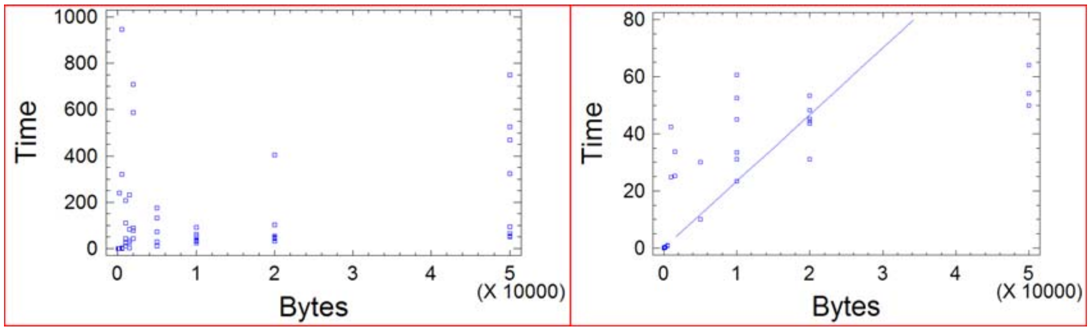

Figure 1. Time it takes to send the bytes over a network without removing delays (left) and eliminating (right).

The result shows that the time is practically lineal with respect to number of bytes, as pointed all hypotheses. That little variation is due to bytes number of header that added distinct existing protocols over network (TCP, IP…). In particular, this model explained a 88,5351% of variability. The correlation coefficient is 0.940931, indicating a relatively strong relation between variables.

Analizing máximum values (Figure 2), we obtain Time ≈ 0,0032 Bytes. The stadistic 'R-square' indicates to model explain 82,7288% of variability on Time, after transformation of logarithmic scale to linearize model. The correlation coefficient is 0.909554, indicating a relatively strong relation between variables.

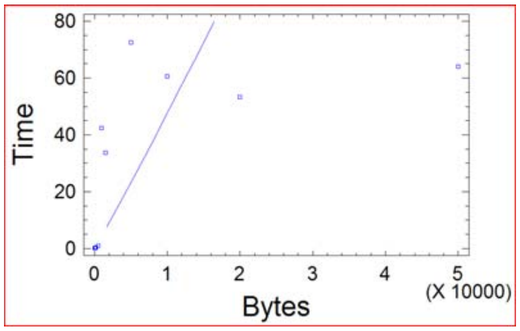

Figure 2. Maximum communication time.

When communication must be done over the same computer, transmission time is null.

Hence, we conclude, to send all equal objects of one membrane to another, it is needed a máximum time (on seconds) approximate:

T_{max} ≈ 0,0256 object

To know the needed time on comunication phase is not enough with multiply the number of distinct objects to send by the time of each one, for each one of the membranes. Experimentally, we obtained values that rapidly change due to the same problems previously mentioned. Taking into account these deviations and the membranes are all connected with the same number of membranes and send the same number of objects, we get:

$$T \approx 58{,}42 + 8{,}16 \text{ Membrane} + 3{,}16 \text{ Object}$$

In [Tejedor, 2007], consider the communication time depends only on the transmission time of bytes and a constant. With this formula from empirical data find that this constant is dependent on the number of membranes that communicate. This is a direct result that the communication phase included the composition of the new multiset of objects, and this is done in each membrane.

Distribution of membranes

Taking the communication times and [Tejedor, 2007] studies, on the same P system and a certain number of processors, execution times depend on the distributions of the membranes are made. In Figure 3 we can see that both distributions present different times in their implementation. If we consider that all processors have three membranes, and consider the execution time of the application phase equal, it will not show differences. Internal communications are negligible with respect to external communications. Just analyze communications between processors.

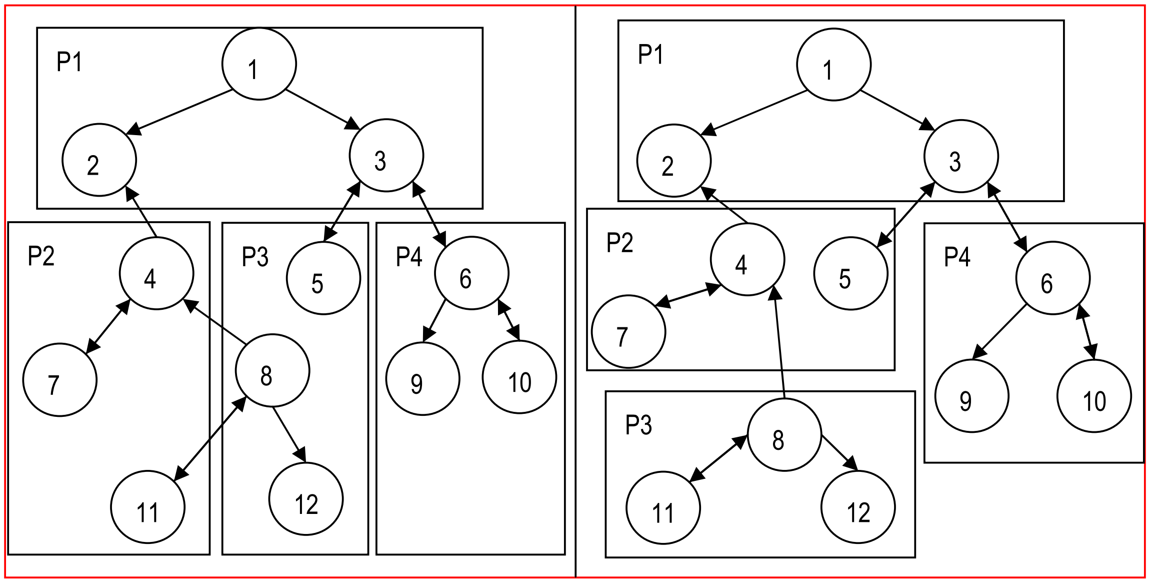

Figure 3. Two distributions of PSystem in four processors. The edges indicate how the symbols are sent from one membrane to another according to the rules.

We assume that communications between membranes only send one objects. The times to communicate will:

P1 -> T = 58,42 +8,16 * 2 +3,16 = 77,90	P1 -> T = 58,42 +8,16 * 2 +3,16 = 77,90
P2 -> T = 58,42 +8,16 * 2 +3,16 = 77,90	P2 -> T = 58,42 +8,16 * 2 +3,16 = 77,90
P3 -> T = 58,42 +8,16 * 3 +3,16 = 86,06	P3 -> T = 58,42 +8,16 * 1 +3,16 = 69,74
P4 -> T = 58,42 +8,16 * 1 +3,16 = 69,74	P4 -> T = 58,42 +8,16 * 1 +3,16 = 69,74

One can see that if messages are sent in parallel, or following a token ring architectures proposed by [Tejedor, 2007] or [Bravo, 2008], time in the second case is less.

Conclusion

It has been shown empirically that the time required for the communication phase will depend on the number of objects that are sent, and the different target membrane. We also observe that these times will be extended by different actors outside the system. Given these times, with a good distribution of the membranes in processor, will reduce communication time, and therefore, the whole times.

There remain many studies to be performed on the communication phase, such as the degree to which they affect a processor to communicate with various, or that the number of symbols is different for each membrane.

Bibliography

[Arroyo, 2003] F. Arroyo, C. Luengo, A.V. Baranda, L.F. de Mingo. A software simulation of transition P System in Haskell. In: Lecture Notes in Computer Science 2597. Springer-Verlag, Berlin 2003, Pp: 19-32.

[Balbotín-Noval, 2003] D. Balbotín-Noval, M.J. Pérez-Jiménez, F. Sancho-Caparrini. A MzScheme Implementation of Transition P System. IN: Membrane Computing, LNCS 2597, Spring Verlag. Berlin 2003. Pp: 57-73.

[Bravo, 2008] G. Bravo, L. Fernández, F. Arroyo, M. Peña. Hierarchical Master-Slave Architecture for Membrane Systems Implementation. In: The 13th International Symposium on Artificial Life and Robotics. 2008. Pp: 485-490.

[Ciobanu, 2004] G. Ciobanu, W. Guo. P System Running on a Cluster of Computers. In: Workshop on Membrane Computing, LNCS 2933. 2004. Pp: 123-2004.

[Cordón-Franco, 2004] A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F. Sancho-Caparrini. A Prolog Simulator for Deterministic P Systems with Active Membranes. In: New Generation Computing, 22(4). 2004. Pp: 349-363.

[Fernández, 2005] L. Fernandez, V.J. Martinez, F. Arroyo, L.F. Mingo, A Hardware Circuit for Selecting Active Rules in Transition P Systems. In: First International Workshop on Theroy and Application of P System. September 2005. Pp: 45-48.

[Gomez, 2007] S. Gómez, L. Fernández, I. García, F. Arroyo. Researching Framework for Simulating/Implementating P Systems. In: Fifth International Conference Information Research and Applications (i.TECH-2007). Varna (Bulgary) june, 2007.

[Malita, 2000] M. Malita. Membrane Computing in Prolog. In: Preproceedings of the Workshop on multiset Processing. 20000. Pp: 159-175.

[Martínez, 2006] V. Martínez, L. Fernández, F. Arroyo, A. Gutiérrez. Implementation HW of an Enclosed Algorithm to Rules Applications in a Transition P System. In: 8-th International Symposium on Syumbolic and Numeric Algorithms for Scientific Computing. 2006.

[Nepomuceno-Chamorro, 2004] I.A. Nepomuceno-Chamorro. A Java Simulator for Basic Transition P System. In: Journal of Universal Computer Science. 2004. Pp: 620-619.

[Paun, 2000] G. Paun. "Computing with membranes". In: Journal of Computer and System Sciences, 1(61). 2000. Pp: 108-143.

[Petreska, 2003] B. Petreska, C. Teuscher. A hardware membrane system. In: Preproceedings of the Workshop on Membrane Computing. 2003. Pp: 242-255.

[Suzuki, 2000] Y. Suzuki, H. Tamaka. On a Lisp implementation of a class of P Systems. Romanian Jounal of Information Science and Technology. 2000. Pp: 173-186.

[Syropoulos, 2003] A. Syropoulos, E.G. Mamatas, P.C. Allilomes, K.T. Sotiriades. A distributed simulation of P Systems. In: Preproceedings of the Workshop on Membrane Computing. 2003. Pp: 455-460.

[Tejedor, 2007] J. Tejedor, L. Fernández, F. Arroyo, G. Bravo:. An architecture for attacking the bottleneck communication in P systems. In: M. Sugisaka, H. Tanaka (eds.), Proceedings of the 12th Int. Symposium on Artificial Life and Robotics. 2007. Pp: 500-505.

Authors' Information

***Miguel Angel Peña** – Dept. Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain; e-mail: m.pena@fi.upm.es*

***Jorge Tejedor** – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: jtejedor@eui.upm.es*

***Juan B. Castellanos** – Dept. Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain; e-mail: jcastellanos@fi.upm.es*

***Ginés Bravo** – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: gines@eui.upm.es*