# Looking for P truth

**A. Cordón-Franco; M.A. Gutiérrez-Naranjo; M.J. Pérez-Jiménez**

Dpto. de Ciencias de la Computación e Inteligencia Artificial
E.T.S. Ingeniería Informática. Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012, Sevilla, España
E-mail: {`acordon, magutier, marper`}`@us.es`

**Abstract.** In a general sense, Logic studies how to derive new pieces of information from previous ones. In this paper we explore the analogies between P systems, where new configurations can be obtained from previous ones by using a set of rules, and the derivation of new theorems from previous ones or from axioms by using inference rules.

## 1 Introduction

In P systems [5], the information about a configuration is stored in multisets associated with a like-tree structure of membranes. A given configuration produces, in a discrete amount of time (a *evolution step*), a new configuration by using a set of rules in parallel maximal mode. Usually, the set of rules can be applied in a non deterministic way; so it is more convenient to consider, instead of a single configuration, the set of all the configurations that can be obtained from a previous one in one step by using the corresponding set of rules of the P system.

This schema of generating new pieces of information from previous ones by using a set of rules has been widely studied both in Logic and in Computer Science. Production systems and systems based on rewriting rules are only examples of systems where *if-then* rules are applied. From a Computer Science point of view, *if-then* rules provide a general framework for several programming paradigms. In fact, we can find specific languages as CLIPS [6] or Prolog [1, 7] where these rules are on the basis of the system of representation and reasoning.

In this paper we initiate the study of the similarities between inference models based upon first–order logic and P systems. We focus on the definition of a fixpoint semantic for P systems. In the literature, a good amount of examples of fixpoint semantics for programming languages can be found and they provide evidence that this kind of semantics constitutes an adequate tool for motivating and justifying methods for obtaining properties of programs.

## References

[1] Bratko, I.: *PROLOG Programming for Artificial Intelligence*, Third edition. Addison-Wesley, 2001.

[2] Cordón-Franco A., Gutiérrez-Naranjo, M.A.,Pérez-Jiménez M.J., Riscos-Núñez and Sancho-Caparrini F.: *A Prolog Simulator for Deterministic P Systems with Active Membranes New Generation Computing*. In press.

[3] Cordón-Franco A., Gutiérrez-Naranjo, M.A.,Pérez-Jiménez M.J., Riscos-Núñez and Sancho-Caparrini F.: *Implementing in Prolog an effective cellular solution to the Knapsack Problem Membrane Computing WMC 2003* Lecture Notes in Computer Science 2933. Springer Verlag, 2004, pp.:140-152

[4] Lloyd J.W.: Foundations of Logic Programming. (2nd ed.) Springer, Berlin, 1987

[5] Păun, G.: *Computing with membranes Journal of Computer and System Sciences*, **61**(1), 2000, 108–143.

[6] CLIPS - A Tool for Building Expert Systems
`http://www.ghg.net/clips/CLIPS.html`

[7] Logic Programming: `http://www.afm.sbu.ac.uk/logic-prog/`