11 Years of P-Lingua: A Backward Glance

Ignacio Pérez-Hurtado, David Orellana-Martín, Miguel Á. Martínez-del-Amor, Luis Valencia-Cabrera, Agustín Riscos-Núñez, Mario J. Pérez-Jiménez

Reseach Group on Natural Computing
Universidad de Sevilla
Avda. Reina Mercedes S/N, 41012 Sevilla, Spain
{perezh,dorellana,mdelamor,lvalencia,ariscosn,marper}@us.es

Summary. In 2008, P-Lingua was born. The Research Group on Natural Computing worked on the development of simulation tools since the beginning of Membrane Computing. However, back in 2007, researchers from the group set out to the ambitious journey of creating a generic simulation framework for P systems. P-Lingua has evolved since then, offering more flexibility and a wider range of supported models. Many applications have also branched from this software project. In this paper, we briefly survey the evolution of P-Lingua to date, some of the associated applications, and prospective paths for upcoming challenges in the research area.

Key words: P systems, Simulation Software, P-Lingua

1 Introduction

Since the dawn of Membrane Computing, software tools to perform P system simulations have been developed. In fact, simulators have accompanied the evolution of P system models [34, 3, 30], given that they are the best way to aid in experimental validation and in formal verification. In the early years of the research area, simulators were conceived for specific models, leading to a plethora of interfaces and platforms, each one very different to each other. In this context, developing an uniform simulation framework was a need. This was the seed that led to P-Lingua project [36], which was the major work presented at Ignacio Pérez-Hurtado's doctoral dissertation [30]. Further theses developed at the core of the Research Group on Natural Computing were carried out with specific contributions to P-Lingua [23, 8, 33, 15, 24].

P-Lingua aims at flexibility and extensibility to simulate P systems. The term itself is employed at different extents, so it is usually confused. P-Lingua can be used to denote:

- *the software project*: P-Lingua is an open-source software project published under GNU GPL license. All the developed code is freely available to the

community, and we hope that it helps other researchers to run their models and to develop new simulators. It includes the parser, the generation of files, the simulation framework.

- *the simulation framework*: more commonly known as pLinguaCore, it is Java framework that aims at simulating P systems in a flexible and extensible way. Several design pattern of object oriented programming were employed to help developing new simulations in a clear and clean manner.
- *the programming language*: instead of having an specific GUI to enter the information of the models, P-Lingua enables P-system designers to write their models into plain-text files in a very similar way (i.e. with similar syntax) than they write them down in paper or whiteboard.
- *the files with .pli extension*: files including models described in P-Lingua programming language have the .pli extension, and they are the input for the simulation software framework.
- *the command-line tool*: pLinguaCore can be employed from the Terminal (Linux) or Cmd (Windows) using a command-line tool with specific parameters to define the input .pli files, the output files (when used as a compiler) or the simulation options (when used as a simulator).
- *the translator/compiler to binary files*: efficient, parallel simulators came after P-Lingua, looking for small runtimes. They harness the P-Lingua parsing engine in order to process .pli files, and specific binary files are generated for them.

In this paper, we look back and survey the evolution of P-Lingua through the 11 years of developments and efforts in the research community. In Section 2, a gentle introduction of the P-Lingua programming language is made. In Section 3, a glance at the evolution of P-Lingua software is provided. Graphical User Interfaces (GUIs) based on P-Lingua are revisited in Section 4. The connection of P-Lingua to parallel simulators is summarized in Section 5. Conclusions and plans to future developments are displayed in Section 6.

2 P-Lingua Description Language

The simulation of P system computations over conventional silico machines has been done from the beginning of Membrane Computing.

The general structure of a simulator should contain three main modules:

- **The input**: A way to define the P system to be simulated as well as the initial configuration.
- **The simulation core**: The implementation of one or more simulation algorithms to reproduce the behavior of the P system to be simulated.
- **The output:** The representation of relevant information to the final user, it could be the representation of the final configuration, one computation, the whole computation tree, etc. In some cases, the information should be processed, developing statistics and providing graphics.

The simulation core usually depends on the software/hardware architecture to be used, highly parallel architectures such as CUDA use simulation algorithms very different than the ones used with approximations such as developing in Java or C/C++. The output depends on the final goal of the simulator and the final user background, for example, one application for simulating ecosystems should show information about populations, environment conditions such as weather, etc. Applications for membrane computing should show the generated computation, multisets of objects, executed rules, etc.

Regardless of the type of application, the definition of the P system is mandatory and it is a common module for all the simulators. One approximation is to *hard-code* the P system definition, but it could lead to errors and it is difficult to debug. Using graphical user interfaces introduces dependencies with the technology and operating systems, using GUIs for the P system definition also implies a maintenance effort and they can become obsolete over the years.

Thus, the solution used in P-Lingua is to use a programming language to define P systems. The files written in P-Lingua are close to the standard scientific notation, minimizing the effort of writing them. P-Lingua code is parametric using variables and iterators, modular using reusable source blocks and it can be translated to other formats by using parsers. We could say that the main goal of the P-Lingua language is to minimize the time from the *blackboard* to the *computer*.

As example, next is a P-Lingua code defining a family of recognizer P systems to solve the SAT problem in the framework of cell-like P systems with active membranes and elementary division rules based on the solution presented in [31]. It shows the main characteristics of the language. An exhaustive description of the language can be found in the literature [4, 5, 22, 30] and it is out of the scope of this review.

```
@model<membrane_division>
def Sat(m,n)
{
/* Initial configuration */
@mu = [[]'2]'1;

/* Initial multisets */
@ms(2) = d{1};

/* Set of rules */
[d{k}]'2 --> +[d{k}]-[d{k}] : 1 <= k <= n;

{
+[x{i,1} --> r{i,1}]'2;
-[nx{i,1} --> r{i,1}]'2;
-[x{i,1} --> #]'2;
+[nx{i,1} --> #]'2;
} : 1 <= i <= m;
```

```
{
+[x{i,j} --> x{i,j-1}]'2;
-[x{i,j} --> x{i,j-1}]'2;
+[nx{i,j} --> nx{i,j-1}]'2;
-[nx{i,j} --> nx{i,j-1}]'2;
} : 1<=i<=m, 2<=j<=n;

{
+[d{k}]'2 --> []d{k};
-[d{k}]'2 --> []d{k};
} : 1<=k<=n;

d{k}[]'2 --> [d{k+1}] : 1<=k<=n-1;
[r{i,k} --> r{i,k+1}]'2 : 1<=i<=m, 1<=k<=2*n-1;
[d{k} --> d{k+1}]'1 : n <= k<= 3*n-3;
[d{3*n-2} --> d{3*n-1},e]'1;
e[]'2 --> +[c{1}];
[d{3*n-1} --> d{3*n}]'1;
[d{k} --> d{k+1}]'1 : 3*n <= k <= 3*n+2*m+2;
+[r{1,2*n}]'2 --> -[]r{1,2*n};
-[r{i,2*n} --> r{i-1,2*n}]'2 : 1<= i <= m;
r{1,2*n}-[]'2 --> +[r{0,2*n}];
-[c{k} --> c{k+1}]'2 : 1<=k<=m;
+[c{m+1}]'2 --> +[]c{m+1};
[c{m+1} --> c{m+2},t]'1;
[t]'1 --> +[]t;
+[c{m+2}]'1 --> -[]Yes;
[d{3*n+2*m+3}]'1 --> +[]No;

} /* End of Sat module */

/* Main module */
def main()
{
/* Call to Sat module for m=4 and n=6 */

call Sat(4,6);

/* Expansion of the input multiset */

@ms(2) += x{1,1}, nx{1,2}, nx{2,2}, x{2,3},
nx{2,4}, x{3,5}, nx{4,6};
```

```
} /* End of main module */
```

3 P-Lingua Simulation Software

PLinguaCore [36] is a GNU GPL library written in Java for parsing P-Lingua files, simulate computations and translate input P-Lingua files to other file formats. The library detects errors in the file and reports them such as a regular compiler tool. For each supported P system variant, several simulation algorithms are included. Eventually, the library translates files, which define a P system, between formats, for instance, from P-Lingua format to binary format.

Each version of the library is associated with an extension of the programming language and simulation engine to cover more types of P systems:

- **pLinguaCore 1.0**: The initial version. It was able to define active membrane P systems with division rules. pLinguaCore was in very early state. [4].
- **pLinguaCore 2.0**: Several cell-like P system models and built-in simulators for each supported model were added [5].
- **pLinguaCore 2.1**: Support for tissue-like P systems with division rules were added [22], plus fixed bugs.
- **pLinguaCore 3.0**: The simulation algorithm called DCBA for Population Dynamics P systems (PDP systems) was added [19], along with a new binary output file for PDP systems [17]. In this version, stochastic P systems are discontinued. Some bugs were fixed.
- **pLinguaCore 4.0**: Support for Spiking Neural P systems was added [13]. Moreover, Tissue-Like P systems with Cell Separation Rules was also added [29]. Some bugs were also fixed.

The development of pLinguaCore continued as part of MeCoSim tool, as depicted in next section.

4 GUIs for P-Lingua

As highlighted from the beginning of this survey paper, from its very beginning P-Lingua aimed to provide a general-purpose framework for the description and simulation of P systems. This would emerge as a greatly valuable tool for P systems experts to work with P systems by specifying them in a similar way that they would do with paper and pen in their theoretical studies, as shown in the previous sections.

However, in order to provide these users with more usable mechanisms to experiment with models based on P systems, they would require more visual elements to help them save time and clarify the evolution of the systems under study.

Additionally, membrane systems proved to be useful as a modelling framework for real problems in areas ranging from biology to economy, and a second group of

interest appeared into the scene: people not necessarily familiar with P systems, but using the models designed by experts to help understanding and managing the phenomena they were interested in. These kinds of *end users* would need higher level tools to handle the systems but not entering into technical details, and here some graphical interfaces would also play an essential role.

Among these visual interfaces to handle P systems based on the P-Lingua framework, it is worth mentioning two especially relevant ones: MeCoSim and MeCoGUI, described within the following subsections.

4.1 MeCoSim

From the very beginning in 2009-2010 [28], `MeCoSim` was designed with the two-fold intention expressed above, to help P systems designers with the modelling and verification tasks, and also allowing end users to handle solutions based on membrane systems [33, 30].

For the first goal, this software was always supported by the parsing, debugging and simulation engine given by `P-Lingua`, while the latter objective required further development to provide some mechanism where P systems designers could easily prepare end-user applications by a relatively simple configuration. Thus, `MeCoSim` was initially designed to enable the user defined customized interfaces, with inputs, outputs, charts, etc., adapted to each model or solution given by a family of P systems, expressed in `MeCoSim` language. This enables the users to enter their data for different initial conditions, instantiating the desired P systems within the family.

Apart from the initial goals, this software environment had from its conception two essential objectives: the flexibility required for the definition of any kind of custom applications and variety of P systems accepted, and the extensibility to increase its initial functionalities through the development using its plugins architecture and the integration with external packages. Thus, the seminal version of the software was enriched by the integration with different tools for the property extraction and verification of P systems [12, 9] and the development of plugins for graph-based problems, definition of propositional formulas, extensions of the language accepted by the generation of parameters, connection with external simulators based on `Spin` [10] and several other new features for designers and end users. Extensive documentation, case studies, explicative videos, etc. about everything related with MeCoSim can be found at [35].

P-Lingua inside MeCoSim

Given the nature of this survey paper, devoted to P-Lingua, it is worth mentioning that MeCoSim is fully compatible with any version of P-Lingua, and uses pLinguaCore as its engine for parsing and simulation (additionally, it allows the simulation through other external simulators, but its most extended use makes use of pLinguaCore). Thus, MeCoSim will mainly expose the types of computing

models, with their parsers and simulators, provided by the version of pLingua we decide to deploy with MeCoSim. Therefore, depending on which version of pLinguaCore (see previous sections) we use, a different set of models and simulators will be available to use in MeCoSim.

However, the scope of MeCoSim and the collaboration with different external use brought new needs in terms of types of models to cover in terms and features of the P-Lingua language itself. The time constraints of P-Lingua project and MeCoSim-based applications used by different research groups were different, and was not possible to make both goals fit together with a sound coordination. Hence, a new version of pLinguaCore, let us say a fork from the project, started to grow inside MeCoSim project, and MeCoSim integrated this new version of pLinguaCore inside its distribution, progressively diverging from pLinguaCore 4. Despite this divergence, however, all models available in pLinguaCore 4 were also available inside pLinguaCore version used by MeCoSim. In this sense, we could informally consider this internal version as some *unofficial* pLinguaCore 4.5, including its predecessor plus some additional features in the language and some further models (with their parsers and simulators).

Among the main new computing models added within P-Lingua version inside MeCoSim we can find:

- Simple kernel P systems[9].
- Probabilistic Guarded (Scripted) P systems[7, 6].
- Evolutional-communication P systems[25].
- Fuzzy Reasoning Spiking Neural P systems[14].
- Cell-like Spiking Neural P systems[32].
- P systems with minimal cooperation.
- Tissue P systems with promoters.

4.2 MeCoGUI

MeCoGUI is a minimal graphical user interface for P-Lingua which simplifies the process of parsing input parameters and generating results files. The current version encapsulates a version of P-Lingua which integrates Probabilistic Guarded Scripted P (PGSP) systems, a novel Membrane Computing framework tailored for modelling population dynamics [8, 7].

PGSP systems have been used to study the behaviour and conservation trends of *Pieris napi oleracea*, (P. n. oleracea, for short). This species, commonly known as mustard white butterfly, is native from eastern North America. The aim of the model is to predict population growth trends and evolutionary responses of several genotypes of this species, considering the concurrent invasions of the exotic plant garlic mustard (*Alliaria petiolata*), and another exotic, *Cardamine pratensis* [6].

4.3 pLinguaPlugin

pLinguaPlugin was an Eclipse plug-in which provided an end-user interface for using pLinguaCore. Hence, all services performed by pLinguaCore, such as com-

pilation of P system files, error detection, file translation and P system simulation were accessed in an intuitive and user-friendly way by means of pLinguaPlugin. It also provided ways to visually display configurations and modify specifications of P systems on real time. This component is free software published under GNU GPL license, and it is only compatible for pLinguaCore 2.0. After that, the support was discontinued. The plugin is still available at the P-Lingua website [36].

4.4 P-Lingua web analyzer

A web analyzer for P-Lingua was developed in PHP and deployed at the P-Lingua website [36]. An user can upload a .pli file, select to either simulate until a halting configuration or a certain number of steps, and by using pLinguaCore as a backend, simulate the model. When the simulation is finished, the output is shown on the website as a tree, where one can select certain configurations, and inside them, depict membranes, objects and rules executed. Moreover, both the simulation and the output data, dumped by pLinguaCore, is shown on the window. This tool was developed as a final project of computer engineering by J. González-Pareja in 2012.

5 Connection to Parallel Simulators on GPUs

pLinguaCore is the software framework devoted to perform P system simulations. As mentioned above, it is developed using object-oriented paradigm and considering design patterns. The objective of it is to offer readability, flexibility and extensibility of the framework for the research community. However, this comes at the cost of performance. Thus, a branch of P-Lingua project was the development of efficient, parallel simulators for P systems. This new project was later codenamed PMCGPU (Parallel simulators for Membrane Computing on the GPU) [37].

The platform employed to accelerate the simulation of P systems was the GPU (Graphics Processing Units), given that, since the introduction of CUDA, it is easily programmable and leverages a high degree of parallelism (from hundreds to thousands of processing cores) [11]. Multicore CPUs were also used to speedup simulations at a lower level through OpenMP. Several models have been simulated on GPUs so far, as surveyed in [16].

In the next subsections, we will show the two CUDA-based simulators that used input files generated by P-Lingua: PCUDA (for active membrane systems) [2] and ABCD-GPU (for PDP systems) [17]. The specific details of this input, binary formats can be consulted in [23].

5.1 Binary files for Active Membranes

The first GPU-based simulator was first presented in the 7th BWMC (2009) [21], focused on recognizer P systems with active membranes and elementary division

(called PCUDA [2]). This was a generic simulator that required a description of the P system to be simulated as input, reproduced only one computation (confluent condition was assumed), and output the description of the configurations in a syntax close to pLinguaCore output module. In this sense, P-Lingua was employed as a compiler: parsing a P-Lingua file describing a P system with active membranes, and generating a compressed binary file with the unwrapped, processed information to be injected to PCUDA simulator. This binary file was carefully designed to compress the amount of bits dedicated to define (in this order): the alphabet, the membrane structure, the rules, and the initial configuration. The header of the file included information about the precision (amount of bits) employed for each part.

pLinguaCore was then extended (version 2.0) to support the generation of a specific binary format devoted to only one P system variant (active membranes). P-Lingua command line was extended distributed alongside the PCUDA simulator [37], so that users were able to generate binary files from .pli files by typing: `java -jar pparser.jar inputFile.pli inputFile.bin`. If the binary file is generated without errors, it can be used as input for the simulator, plus providing some extra information: number of membranes expected to be generated by the model (upper bound), number of objects (symbols) defined in the alphabet, number of threads to be launched on the GPU for each membrane.

5.2 Binary files for PDP systems

PDP systems were also considered for parallel simulators, given the real requirement of efficiency by the applications (named ABCD-GPU). The first version of ABCD-GPU was implemented in OpenMP to harness multicore CPUs [18], and was later extended for GPUs [20]. However, these initial versions aimed at testing the concept of executing DCBA algorithm [19] in parallel, and in order to stress the simulators, randomly generated PDP systems were used as input. They were created inside the simulator, so there was no need to input files. In [17], ABCD-GPU was extended with a input module for binary files. The binary format design follows the same concept than the one used for PCUDA. A header helped to establish the precision (bits) used for each part.

The full support of binary file generation for PDP systems came with pLinguaCore 3.0. The command line was again extended, and the user could parse .pli files and generate binaries by: `java -jar pLinguaCore.jar plingua inputFile.pli -bin inputFile.bin`. If the binary is generated without errors, the file can be used to feed ABCD-GPU, plus indicating the following parameters: number of simulations to run, number of time steps to simulate, number of steps per cycle, and amount of cores to be used for the OpenMP version (when employed).

6 Conclusions and Future Work

P-Lingua is a software project devoted to provide tools to simulate P systems. Developing and maintaining this flexible project is of huge complexity. The wide variety of membrane systems is as large as the imagination of Membrane Computing researchers, and normally some models differ from others in many aspects. Gathering all together under the same software umbrella is perhaps not realizable, but using all the techniques and methods of software engineering might help to other developers. P-Lingua can be seen therefore not as a all-in-one software, but as a key stone to ease the way of developing new simulators.

Future version of P-Lingua are under development, and will be focused not only on flexibility, but also on efficiency. P-Lingua 5 [27] is being re-engineered from scratch using C++ object-oriented programming language, that will help to achieve lower runtimes when dealing with large P-Lingua files. This will open new scalable opportunities for next challenges in Membrane Computing.

P-Lingua is very alive, and we look forward to collaborations in order to extend and evolve it.

Acknowledgments

The authors acknowledge the support from the research project TIN2017-89842-P (MABICAP), cofinanced by "Ministerio de Economa, Industria y Competitividad" (MINECO) of Spain, through the "Agencia Estatal de Investigacin" (AEI), and by "Fondo Europeo de Desarrollo Regional" (FEDER) of the European Union.

References

1. M. Cardona, M.A. Colomer, A. Margalida, A. Palau, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy. A computational modeling for real ecosystems based on P systems, *Natural Computing*, **10**, 1 (2011), 39–53.
2. J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez. Simulation of P systems with Active Membranes on CUDA, *Briefings in Bioinformatics*, **11**, 3 (2010), 313–322.
3. D. Díaz-Pernil, C. Graciani, M.A. Gutiérrez-Naranjo, I. Pérez-Hurtado, M.J. Pérez-Jiménez. Software for P systems. In Gh. Păun, G. Rozenberg, A. Salomaa, editors, *The Oxford Handbook of Membrane Computing*, Oxford University Press, Chapter 17, pages 437–454, 2009.
4. D. Díaz-Pernil, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez. A P-lingua programming environment for Membrane Computing, *Lecture Notes in Computer Science*, **5391** (2009), 187-203.
5. M. García-Quismondo, R. Gutiérrez-Escudero, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez. An overview of P-Lingua 2.0. *Lecture Notes in Computer Science*, **5957** (2010), 264-288.

6. M. García-Quismondo, M.J. Reed, F.S. Chew, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez. Evolutionary response of a native butterfly to concurrent plant invasions: simulation of population dynamics. *Ecological modelling*, **360** (2017), 410-424.
7. M. García-Quismondo, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez. Probabilistic guarded P systems: A formal definition. *Proceedings of the Twelfth Brainstorming Week on Membrane Computing*, Fnix Editora, 2014, pp. 183-206.
8. M. García-Quismondo. *Modelling and simulation of real-life phenomena in Membrane Computing*. Ph.D. Thesis, University of Seville, 2014. `http://hdl.handle.net/11441/66147`
9. M. Gheorghe, F. Ipate, R. Lefticaru, M.J. Prez-Jimnez, A. Turcanu, L. Valencia, M. Garca-Quismondo, F. Mierla. 3-COL problem modelling using simple Kernel P systems. *International Journal of Computer Mathematics*, **90**, 4 (2013), 816-830.
10. F. Ipate, R. Lefticaru, L. Mierla, L. Valencia, H. Hang, G. Zhang, C. Dragomir, M.J. Prez-Jimnez, M. Gheorghe. Kernel P systems: Applications and Implementations. *Advances in Intelligent Systems and Computing*, Volume **212** (2013), 1081-1089
11. D. Kirk, W. Hwu. *Programming Massively Parallel Processors: A Hands On Approach*, Morgan Kauffman, 2010.
12. R. Lefticaru, F. Ipate, L. Valencia, A. Turcanu, C. Tudose, M. Gheorghe, M.J. Pérez-Jiménez, I.M. Niculescu, C. Dragomir. Towards an integrated approach for model simulation, property extraction and verification P systems. In M.A. Martínez, Gh. Păun, I. Pérez, F.J. Romero (eds.) *Proc. of the Tenth Brainstorming Week on Membrane Computing*, vol. I, 2012, pp. 291-318.
13. L.F. Macías-Ramos, I. Pérez-Hurtado, M. García-Quismondo, L. Valencia-Cabrera, M.J. Pérez-Jiménez, A. Riscos-Núñez. A P-Lingua based simulator for Spiking Neural P systems, *Lecture Notes in Computer Science*, **7184** (2012), 257-281.
14. L.F. Macías-Ramos, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez. Simulating FRSN P systems with real numbers in P-Lingua on sequential and CUDA platforms. *Lecture Notes in Computer Science*, **9504** (2015), 262-276.
15. L.F. Macías-Ramos. *Developing efficient simulators for cell machines*. Ph.D. Thesis, University of Seville, 2016. `http://hdl.handle.net/11441/36828`
16. M.A. Martínez-del-Amor, M. García-Quismondo, L.F. Macías-Ramos, L. Valencia-Cabrera, A. Riscos-Núñez, M.J. Pérez-Jiménez. Simulating P systems on GPU devices: a survey. *Fundamenta Informaticae*, **136**, 3 (2015), 269–284.
17. M.A. Martínez-del-Amor, L.F. Macías-Ramos, L. Valencia-Cabrera, M.J. Pérez-Jiménez. Parallel simulation of Population Dynamics P systems: updates and roadmap. *Natural Computing*, **15**, 4 (2016), 565–573.
18. M.A. Martínez-del-Amor, I. Karlin, R.E. Jensen, M.J. Pérez-Jiménez, A.C. Elster. Parallel simulation of probabilistic P systems on multicore platforms. In M. García, L.F. Macías, Gh. Păun, L. Valencia (eds.), *Proc. of the Tenth Brainstorming Week on Membrane Computing (BWMC 2012)*, vol. II, 2012, pp. 17–26.
19. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M. García-Quismondo, L.F. Macías-Ramos, L. Valencia-Cabrera, A. Romero-Jiménez, C. Graciani, A. Riscos-Núñez, M.A. Colomer, M.J. Pérez-Jiménez. DCBA: Simulating population dynamics P systems with proportional objects distribution, *Lecture Notes in Computer Science*, **7762** (2013), 257–276.
20. M.A. Martínez-del-Amor, I. Pérez-Hurtado, A. Gastalver-Rubio, A.C. Elster, M.J. Pérez-Jiménez. Population Dynamics P systems on CUDA, *Lecture Notes in Bioinformatics*, **7605** (2012), 247–266.

21. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, J.M. Cecilia, G.D. Guerrero, J.M. García. Simulation of recognizer P systems by using manycore GPUs. *Proceedings of the Seventh Brainstorming Week on Membrane Computing*, **Volume II**, Seville (Spain), 2009, pp. 45–58.
22. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez. A P-Lingua based simulator for Tissue P systems, *Journal of Logic and Algebraic Programming*, **79**, 6 (2010), 374-382.
23. M.A. Martínez-del-Amor. *Accelerating Membrane Systems Simulators using High Performance Computing with GPU*, Ph.D. Thesis, University of Seville, 2013. `http://hdl.handle.net/11441/15644`
24. D. Orellana-Martín. *The P vs NP problem. Development of new techniques through bio-inspired computing modules*. Ph.D. Thesis, University of Seville, 2019. `https://hdl.handle.net/11441/85318`
25. L. Pan, B. Song, L. Valencia-Cabrera, M.J. Pérez-Jiménez. The computational complexity of tissue P systems with evolutional symport/antiport rules. *Complexity*, Volume 2018, Article ID 3745210, 21 pages
26. Gh. Păun. Computing with membranes, *Journal of Computer and System Sciences*, **61**, 1 (2000), 108–143, and Turku Center for Computer Science-TUCS Report No 208..
27. I. Pérez-Hurtado, D. Orellana-Martín, G. Zhang, M.J. Pérez-Jiménez. P-Lingua in two steps: flexibility and efficiency. *Journal of Membrane Computing*, to appear.
28. I. Pérez-Hurtado, L. Valencia-Cabrera, M.J. Pérez-Jiménez, M.A. Colomer, A. Riscos-Núñez. MeCoSim: A general purpose software tool for simulating biological phenomena by means of P systems. In K. Li, Z. Tang, R. Li, A.K. Nagar, R. Thamburaj (eds.), *IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2010)*, vol. I, 2010, pp. 637–643.
29. I. Pérez-Hurtado, L. Valencia-Cabrera, J.M. Chacón, A. Riscos-Núñez, M.J. Pérez-Jiménez. A P-Lingua based Simulator for Tissue P Systems with Cell Separation, *Romanian Journal of Information Science and Technology*, **17** , 1 (2014), 89-102.
30. I. Pérez-Hurtado. *Desarrollo y aplicaciones de un entorno de programación para Computación Celular: P-Lingua*. Ph.D. Thesis. University of Seville, 2010. `http://hdl.handle.net/11441/66241`
31. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2003, 2 (3), 265 – 285.
32. L. Valencia-Cabrera, T. Wu, Z. Zhang, L. Pan, M.J. Pérez-Jiménez. A simulation software tool for cell-like spiking neural P systems. *Romanian Journal of Information Science and Technology*, **20**, 1 (2017), 71-84.
33. L. Valencia-Cabrera. *An environment for virtual experimentation with computational models based on P systems*. Ph.D. Thesis. University of Seville, 2015. `http://hdl.handle.net/11441/45362`
34. L. Valencia-Cabrera, D. Orellana-Martín, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez. From Super-cells to Robotic Swarms: Two Decades of Evolution in the Simulation of P Systems. *Bulletin of the International Membrane Computing Society*, Number 4, December 2017, 65-87.
35. MeCoSim website. `http://www.p-lingua.org/mecosim`
36. The P-Lingua website. `http://www.p-lingua.org`
37. The PMCGPU project website. `http://sourceforge.net/p/pmcgpu`