Introduction to P-Lingua

Ignacio Pérez Hurtado

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Seville

September 5, 2011. Osuna, Sevilla, Spain

A short presentation of me...

Ignacio Pérez-Hurtado (perezh@us.es)

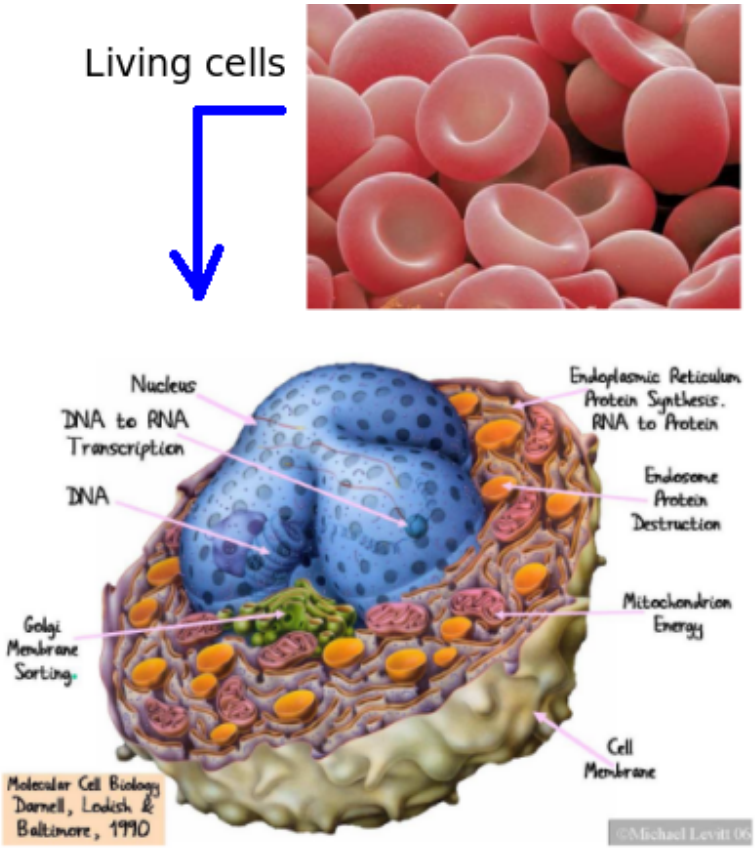
- PhD on Computer Science
  - Desarrollo y Aplicaciones de un Entorno de Programación para Computación Celular: P-Lingua
- Assistant Professor
- Postdoctoral researcher
  - Proyecto de Excelencia de la Junta de Andalucía con Investigador de Reconocida Valía: P08-TIC-04200
- Research Group on Natural Computing
- Dpt. Computer Science and Artificial Intelligence
- University of Sevilla

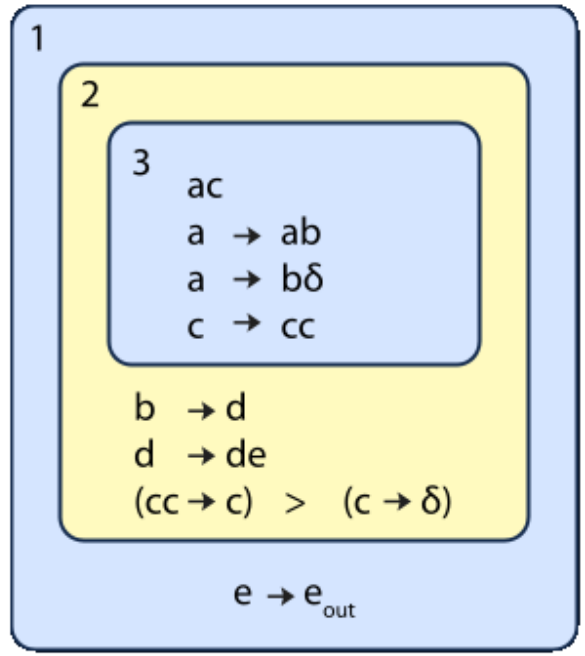
A short presentation of me...

Research interests

- Unconventional Models of Computation
- Membrane Computing
- Bioinformatics
- Systems & Synthetic Biology
- Software Development
- Artificial Intelligence

Membrane Computing





P system

Simulation vs Implementation

- P systems have not been implemented yet
- It is necessary software/hardware to simulate P system computations

Applications of simulators

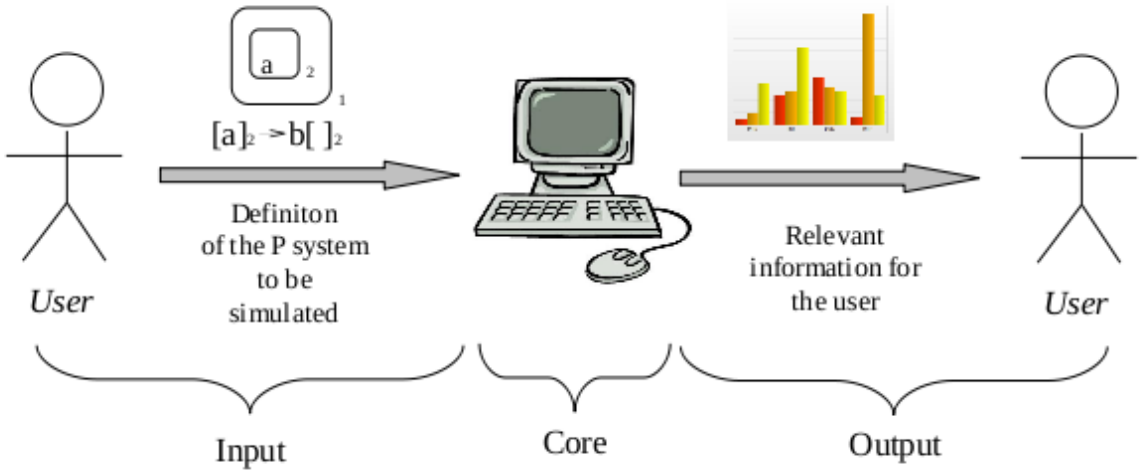
- Informative and educational tools
- Support researching in Membrane Computing
- Simulation, validation and virtual experimentation over models of real-life phenomena

Simulators for P systems

General structure

- Many developed simulators [a]
- Similar structure

[a] Software for P systems. D Díaz Pernil et al. The Oxford Handbook of Membrane Computing. 2010, pp. 437–454



A software framework for Membrane Computing

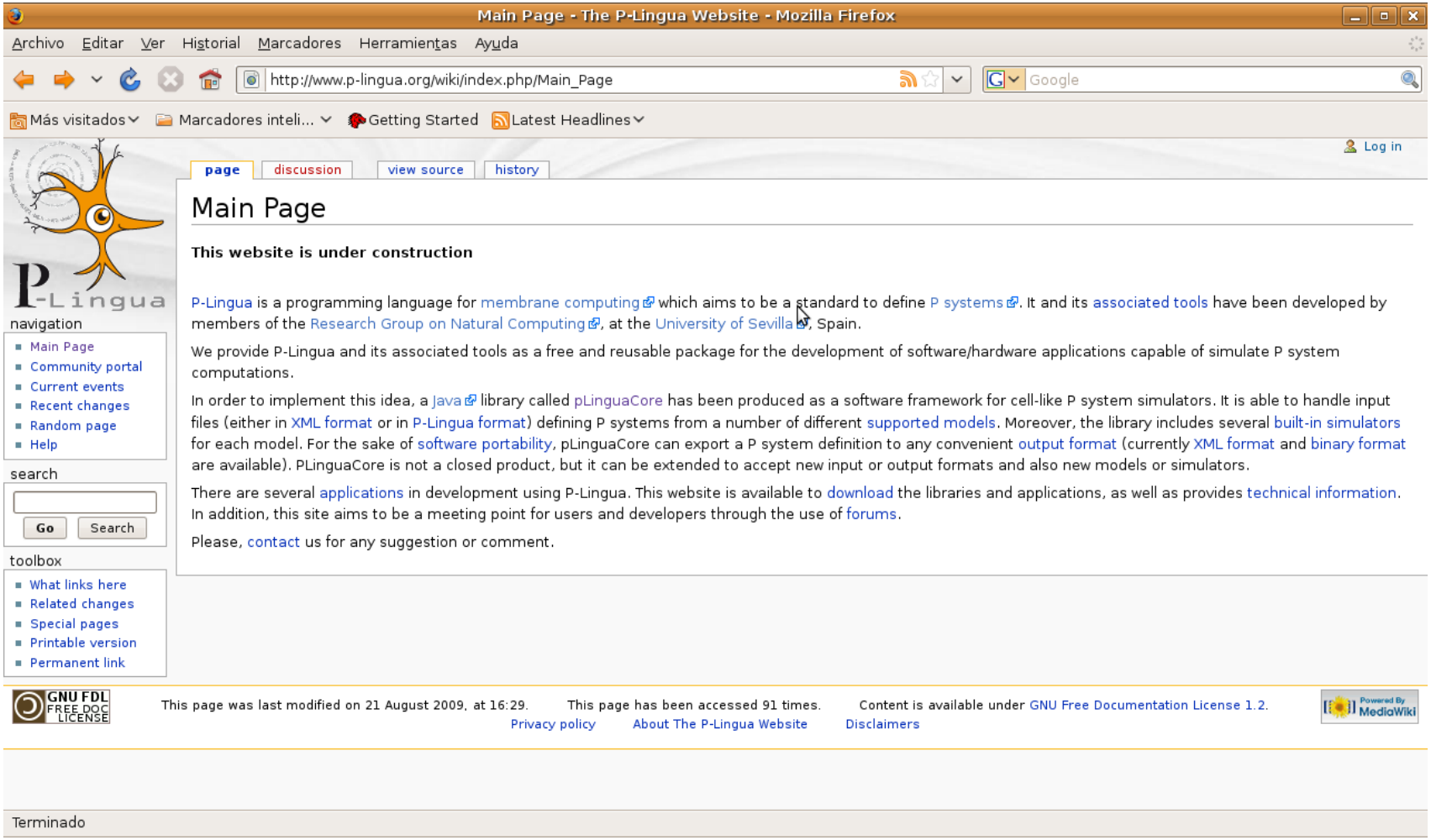
- P-Lingua: A programming language to define P systems
- Tools for compilation and simulation
- pLinguaCore: Library implementing simulation algorithms

P-Lingua: A language to define P systems

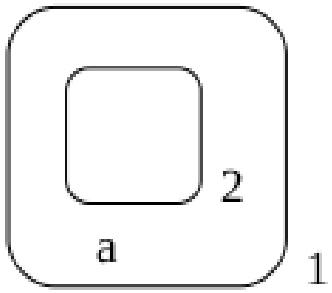
- Language close to scientific notation
- Standar, modular and parametric
- Desacoupled from its applications
- Several supported variants of P systems *cell–like* and *tissue–like*
- Extensible
- Website: `http://www.p-lingua.org`

P-Lingua: A language to define P systems

Página web: `http://www.p-lingua.org`



Example 1: Active membranes with division rules



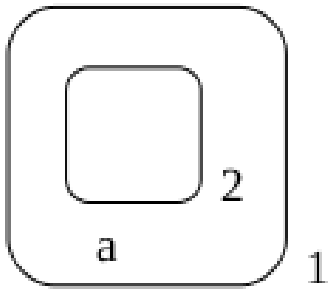
$[a \longrightarrow a, b]_1$

$b[\,]_2 \longrightarrow [c]_2^+$

$[c]_2^+ \longrightarrow [d]_2\ [e]_2^-$

```
@model<membrane_division>
def main()
{
 @mu = [[]'2]'1;
 @ms(1) = a;
 [a --> a,b]'1;
 b[]'2 --> +[c]'2;
 +[c]'2 --> [d]'2 -[e]'2;
}
```

Example 2: Active membranes with creation rules



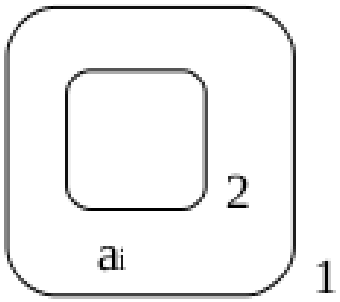
$[a \longrightarrow a, b]_1$

$b[\,]_2 \longrightarrow [c]_2^+$

$[c]_2^+ \longrightarrow [\,[d^2]_3^-\,]_2$

```
@model<membrane_creation>
def main()
{
 @mu = [[]'2]'1;
 @ms(1) = a;
 [a --> a,b]'1;
 b[]'2 --> +[c]'2;
 +[c]'2 --> [-[d*2]'3]'2;
}
```

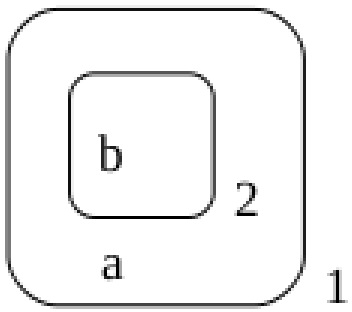
Example 3: Transition P systems



$$[a_i\ [\ ]_2]_1 \longrightarrow [a_{i+1}\ [b_i]_2]_1\ 1 \leq i \leq 10$$

```
@model<transition>
def main()
{
 @mu = [[]'2]'1;
 @ms(1) = a{1};
 [a{i} []'2]'1 --> [a{i+1} [b{i}]'2]'1 : 1<=i<=10;
}
```

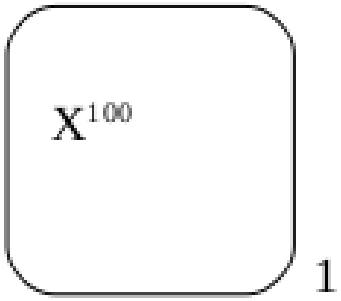
Example 4: Symport/antiport P systems



$a[b]_2 \longrightarrow b[a]_2$

```
@model<symport_antiport>
def main()
{
 @mu = [[]'2]'1;
 @ms(1) = a;
 @ms(2) = b;
 a[b]'2 --> b[a]'2;
}
```

Example 5: Stochastic P systems

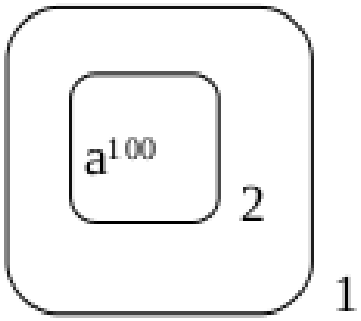


$$[X]_1 \xrightarrow{\lambda} [X^2]_1$$
$$[X]_1 \xrightarrow{\mu} [\,]_1$$

$$\lambda = 1 \quad \mu = 1{,}1$$

```
@variant<stochastic>
def BirthDeath(lambda,mu,nX)
{
 @mu = []'1;
 @ms(1) = X*nX;
 [X]'1 --> [X*2]'1::lambda;
 [X]'1 --> [#]'1::mu;
}
def main() {
 call BirthDeath(1,1.1,100);
}
```

Example 6: Probabilistic P systems



$$[a]_2 \xrightarrow{0,4} b[c]_2$$
$$[a]_2 \xrightarrow{0,4} [d]_2$$
$$[a]_2 \xrightarrow{0,2} e[\,]_2$$

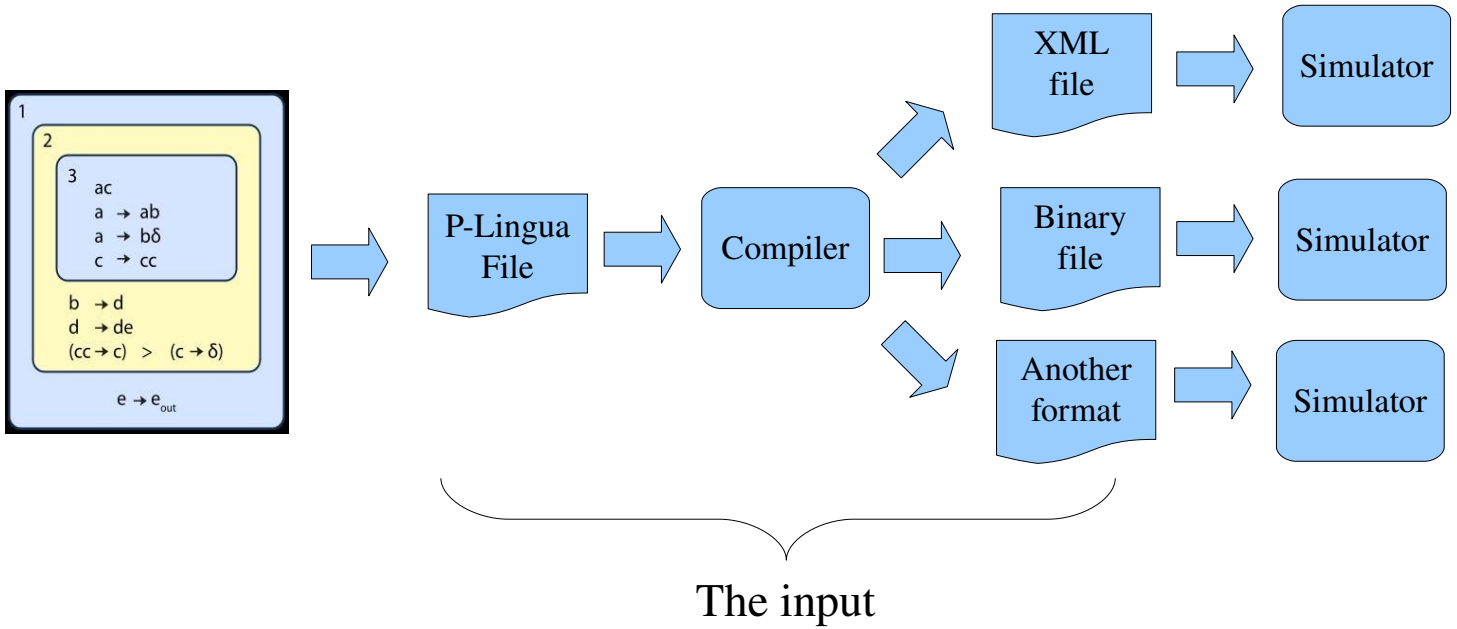
```
@model<probabilistic>
def main()
{
 @mu = [[]'2]'1;
 @ms(2) = a*100;
 [a]'2 --> b[c]'2:: 0.4;
 [a]'2 --> [d]'2 :: 0.4;
 [a]'2 --> e[]'2 :: 0.2;
}
```

Example 7: tissue P systems

```
@model<tissue_psystems>
def main()
{
 @mu = [[]'1 []'2]'0;
 @ms(0) = a;
 @ms(1) = b*5;
 @ms(2) = c*10,d;
 [b]'1 <--> [c*2]'2;
 [c]'1 <--> [a]'0;
 [d]'2  --> [e]'2 [f]'2;
}
```

Command-line compilation tool

Interoperability



Command-line compilation tool

- It checks possible programming errors
- It locates errors on the files

Example: A division rule in “membrane creation”

```
Semantics error: The rule doesn't match
 the "membrane_creation" specification
 in line 38 : 2--28
 Division rules are not allowed
```

pLinguaCore

Software library for simulators

- Java library to implement simulators
- Free software (GNU GPL license)
- It reads P-Lingua files
- It implements several simulation algoritmhs
- It exports to other file formats
- Text interface
- It can be used in other Java applications
- It can be extended
- Web page: `http://www.p-lingua.org`

pLinguaCore

Source code example

```
FileInputStream stream = new FileInputStream("sat.pli");
AbstractParserFactory pf = new InputParserFactory();
InputParser parser =
   (InputParser) pf.createParser("P-Lingua");
parser.setVerbosityLevel(5);
Psystem ps = parser.parse(stream);
ISimulator sim =
   ps.createSimulator(false, false, "active_membranes");
sim.setTimed(true);
sim.setVerbosity(1);
sim.run();
```

Current state of the art

- Several P system models supported
- The last one: Spiking Neural P systems (CMC12)
- Several algorithms implementations
- The last one: A new algorithm for Probabilistc P systems
- Two end-degree projects in course
  - A compiler based on AntLR (September)
  - A Web Interface (December)

Open problems

- A more generical syntax
  - Can we define P systems by ingredients?
- Is it possible a generical simulator?
- More powerful compilers
  - Can we generate optimized source code (I.e. in C/C++)?
- More efficient simulators: HPC, GPUs, FPGAs...

Thank you!