Writing Membrane Systems in P-Lingua 5

Ignacio Pérez Hurtado de Mendoza

Research Group on Natural Computing
Dpt. of Computer Science and Artificial Intelligence
Universidad de Sevilla, Spain

19th Brainstorming Week on Membrane Computing

Seville, January 24–27, 2023

Outline

Membrane computing

Simulators in membrane computing

How to define the input of the simulator?

The problem with new P system variants

Solving the problem

Conclusions, future/present work and references

Membrane computing

Simulators in membrane computing

How to define the input of the simulator?

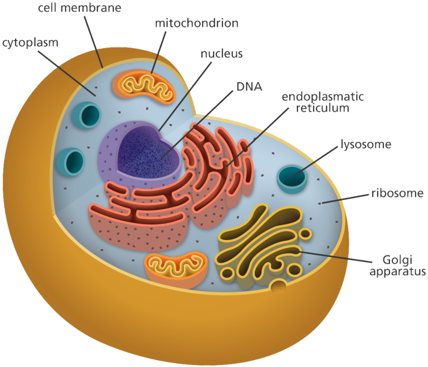
The problem with new P system variants

Solving the problem

Conclusions, future/present work and references

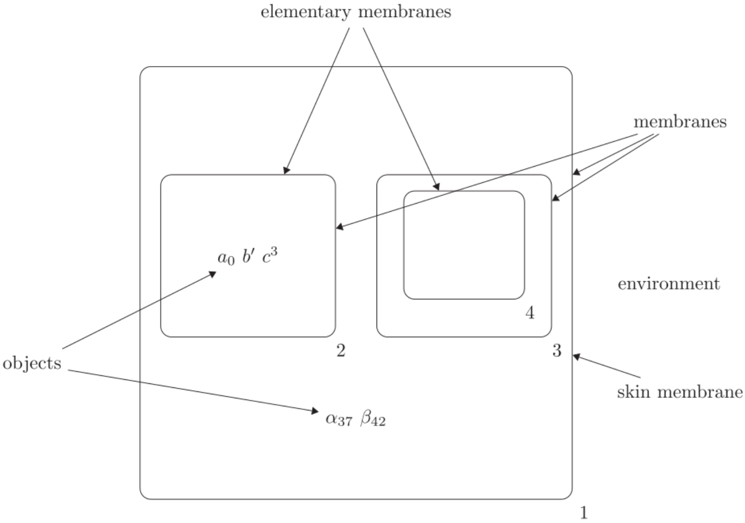
Membrane computing

- Branch of natural computing
- Inspired in the structure and functions of the living cells



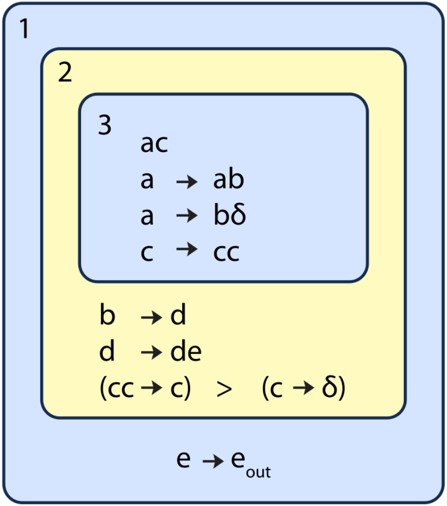
Computational devices in membrane computing

Membrane systems or P systems



Example

Computing squared numbers



1

2

3

ac

$a \rightarrow ab$

$a \rightarrow b\delta$

$c \rightarrow cc$

$b \rightarrow d$

$d \rightarrow de$

$(cc \rightarrow c) > (c \rightarrow \delta)$

$e \rightarrow e_{out}$

Syntax and semantics of membrane systems

- Syntax
  - Definition of the initial structure
  - Definition of initial multisets
  - Definition of rules

- Semantics
  - How rules are selected and executed
  - Described by derivation modes

- A P system *variant* includes the permitted syntactic ingredients for a type of P systems together with the definition of its semantics

Membrane computing

Simulators in membrane computing

How to define the input of the simulator?

The problem with new P system variants

Solving the problem

Conclusions, future/present work and references

Simulators in membrane computing

Motivation

Simulation vs Implementation

- Membrane systems have not been implemented yet[1]
- We need software (or hardware) to simulate computations

Applications of simulators

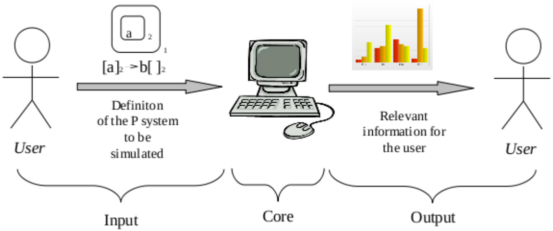
- Pedagogic tools
- Tools to assist researchers in membrane computing
- Simulation, validation and virtual experimentation of models based of membrane computing

[1] Maybe this is a controversial affirmation, but this is a provocative speak!

Simulators in membrane computing

General structure

- There is a wide variety of simulation tools [2]
- They (usually) have a general structure


a
2
1
[a]2 -> b[ ]2
Definiton of the P system to be simulated
User
Relevant information for the user
User
Input
Core
Output

[2] L. Valencia-Cabrera et al. An interactive timeline of simulators in membrane computing. *Journal of Membrane Computing* **1**, 209–222 (2019)

Membrane computing

Simulators in membrane computing

How to define the input of the simulator?

The problem with new P system variants

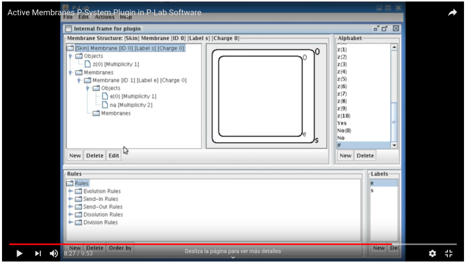
Solving the problem

Conclusions, future/present work and references

Let's begin by the beginning

How to define the input of the simulator?

- Definition of the p system to be simulated
  - Initial membrane structure
  - Initial multisets
  - Set of rules
- Our first approach was a GUI called P-Lab[3]



Active Membranes P-System Plugin in P-Lab Software

Internal frame for plugin

Membrane Structure: [Skin] Membrane [ID 0] [Label s] [Charge 0]

```
[Skin] Membrane [ID 0] [Label s] [Charge 0]
  Objects
    z(0) [Multiplicity 1]
  Membranes
    Membrane [ID 1] [Label e] [Charge 0]
      Objects
        e(0) [Multiplicity 1]
        na [Multiplicity 2]
      Membranes
```

Alphabet: z(1), z(2), z(3), z(4), z(5), z(6), z(7), z(8), z(9), z(10), Yes, No(0), No, #

```
Rules
  Evolution Rules
  Send-In Rules
  Send-Out Rules
  Disolution Rules
  Division Rules
```

Labels: e, s

[3] https://www.youtube.com/watch?v=LFHxSPD9Y5M&t=315s

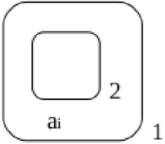
How to define the input of the simulator?

- P-Lab fails since (in our opinion) a GUI to define P systems:
  - could be too rigid
  - could be difficult to extend
  - could be obsolete
- So, why not to move to a definition language?

P-Lingua: A language to define P systems

- Language close to scientific notation
- Standard, modular and parametric
- Decoupled from its applications
- Several supported variants of P systems
- Extensible
- Website: `http://www.p-lingua.org`
- It was presented in the 6th BWMC (2008)

An example: Transition P systems



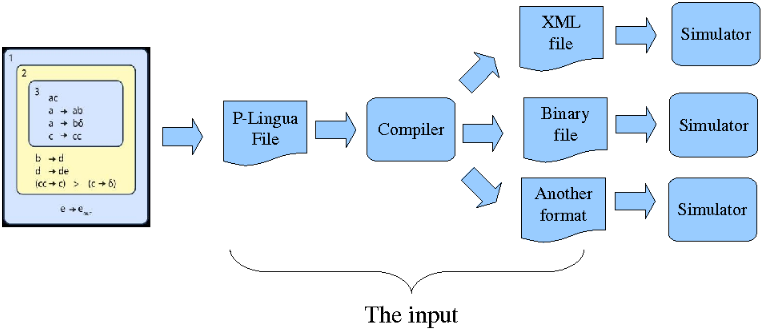
$$[a_i\ [\ ]_2]_1 \longrightarrow [a_{i+1}\ [b_i]_2]_1\ 1 \leq i \leq 10$$

```
@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;
}
```

pLinguaCore

- Java library for compilers and simulators
- Free software (GNU GPL license)
- It reads P-Lingua files
- It generates P system definitions in other formats
- It implements several simulation algorithms

pLinguaCore



The history

- P-Lingua + pLinguaCore 1.0 (2008)
  - Active membranes P systems with division rules
- P-Lingua + pLinguaCore 2.0 (2010)
  - Several cell-like P system variants
  - Built-in simulators
- P-Lingua + pLinguaCore 2.1 (2010)
  - Tissue-like P systems with division rules
- P-Lingua + pLinguaCore 3.0 (2013)
  - PDP systems
  - Several simulators for PDP systems
- P-Lingua + pLinguaCore 4.0 (2014)
  - Spiking Neural P systems
  - Tissue-like P systems with cell-separation rules

Some related tools

- PMCGPU project
  - Moving to efficiency
  - Parallel simulators for MC on the GPU
  - The input is generated with P-Lingua
- MeCoSim: Membrane Computing Simulator
  - A software to design end-user applications based on membrane computing
  - It includes a custom version of pLinguaCore
  - It has been used for simulation of real ecosystems
- And more!

Membrane computing

Simulators in membrane computing

How to define the input of the simulator?

The problem with new P system variants

Solving the problem

Conclusions, future/present work and references

The problem with new P system variants

- Extending pLinguaCore for a new P system variant:
  - Decide a new name to identify the variant.
  - Implement code to extend the parser in pLinguaCore.
  - Implement code to generate custom output formats.
  - Implement one or more simulation algorithms.
- All is hard-coded!
- How to use a variant not supported in pLinguaCore?

Membrane computing

Simulators in membrane computing

How to define the input of the simulator?

The problem with new P system variants

Solving the problem

Conclusions, future/present work and references

Solving the problem

Extend P-Lingua to write P system variants

- P-Lingua 5 includes an extension of the language to define P system variants
- It has retro-compatibility with P-Lingua 4
- pLinguaCore is not longer required
- A new toolkit developed from scratch in C/C++ [4]
    - A parser for the command-line:
        - Input: P-Lingua 5 files
        - Output: P system definition in XML, JSON or binary format
    - A C++ generic simulator for the command-line

[4] I-Pérez-Hurtado et al. A new P-Lingua toolkit for agile development in membrane computing, *Information Sciences* (2022), **587**, 1–22

P-Lingua 5

Example: Cell-Like P systems with membrane division rules

```
!dam_evolution {
   ?[a -> v]’h;
   ?[a ->  ]’h;
}
!dam_send_in {
   a ?[ ]’h -> ?[b]’h;
}
!dam_send_out {
   ?[a]’h -> b ?[ ]’h;
}
!dam_dissolution {
   ?[a]’h -> b;
   ?[a]’h ->  ;
}
```

P-Lingua 5

Example: Cell-Like P systems with membrane division rules

```
!dam_division {
   ?[a]’h -> ?[ ]’h ?[ ]’h;
   ?[a]’h -> ?[b]’h ?[ ]’h;
   ?[a]’h -> ?[ ]’h ?[b]’h;
   ?[a]’h -> ?[b]’h ?[c]’h;
}

@model(membrane_division) =
       dam_evolution,
       // evolution rules are maximally parallel
       @1{dam_send_in, dam_send_out, dam_dissolution, dam_division};
       // upper-bound for send_in, send_out, dissolution, division is 1
```

P-Lingua 5

Example: Cell-Like P systems with membrane division rules

```
@model<membrane_division>
@include "membrane_division_model.pli"
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;
```

P-Lingua 5

Example: Transition P systems

```
!transition_evolution /* Limited to rules with 3 inner membranes */
{
       [a -> v]’h;
       [a -> v, @d]’h;
   (?) [a -> v]’h;
   (?) [a -> v, @d]’h;
       [a [ ]’h1 --> v [w]’h1]’h;
       [a [ ]’h1 --> v [w]’h1]’h;
   (?) [a [ ]’h1 --> v [w]’h1]’h;
   (?) [a [ ]’h1 --> v [w]’h1]’h;
       [a [ ]’h1 [ ]’h2 --> v [w1]’h1 [w2]’h2]’h;
       [a [ ]’h1 [ ]’h2 --> v [w1]’h1 [w2]’h2]’h;
   (?) [a [ ]’h1 [ ]’h2 --> v [w1]’h1 [w2]’h2]’h;
   (?) [a [ ]’h1 [ ]’h2 --> v [w1]’h1 [w2]’h2]’h;
       [a [ ]’h1 [ ]’h2 [ ]’h3 --> v [w1]’h1 [w2]’h2 [w3]’h3]’h;
       [a [ ]’h1 [ ]’h2 [ ]’h3 --> v [w1]’h1 [w2]’h2 [w3]’h3]’h;
   (?) [a [ ]’h1 [ ]’h2 [ ]’h3 --> v [w1]’h1 [w2]’h2 [w3]’h3]’h;
   (?) [a [ ]’h1 [ ]’h2 [ ]’h3 --> v [w1]’h1 [w2]’h2 [w3]’h3]’h;
} @model(transition) = transition_evolution;
```

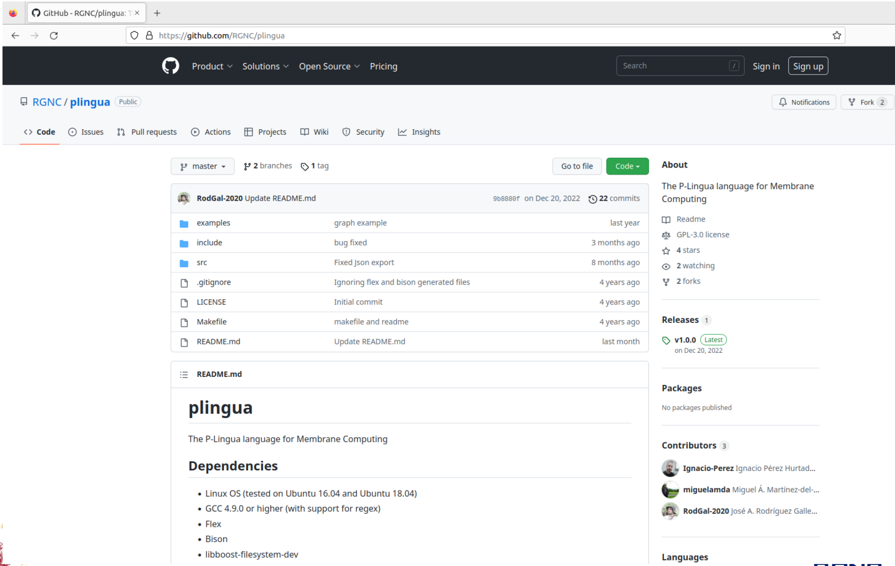
P-Lingua 5

The command-line simulator

- A command-line simulator has been written in C++
- It reads the output generated by the P-Lingua compiler (XML/Json/binary file defining the P system)
- It optionally reads a file with the initial configuration
- It simulates the P system following the defined semantics in the file
- It outputs one computation until a halting state or a number of simulation steps
- It can be run in a non-randomized mode, where it outputs always the same computation for a given P system
- The final configuration is written to a file, the simulation can be re-started

P-Lingua 5

https://github.com/RGNC/plingua



RodGal-2020 Update README.md 9b8880f on Dec 20, 2022 22 commits

examples	graph example	last year
include	bug fixed	3 months ago
src	Fixed Json export	8 months ago
.gitignore	Ignoring flex and bison generated files	4 years ago
LICENSE	Initial commit	4 years ago
Makefile	makefile and readme	4 years ago
README.md	Update README.md	last month

README.md

plingua

The P-Lingua language for Membrane Computing

Dependencies

- Linux OS (tested on Ubuntu 16.04 and Ubuntu 18.04)
- GCC 4.9.0 or higher (with support for regex)
- Flex
- Bison
- libboost-filesystem-dev

About

The P-Lingua language for Membrane Computing

Readme
GPL-3.0 license
4 stars
2 watching
2 forks

Releases 1

v1.0.0 Latest
on Dec 20, 2022

Packages

No packages published

Contributors 3

Ignacio-Perez Ignacio Pérez Hurtad...
miguelamda Miguel Á. Martínez-del-...
RodGal-2020 José A. Rodríguez Galle...

Languages

Membrane computing

Simulators in membrane computing

How to define the input of the simulator?

The problem with new P system variants

Solving the problem

Conclusions, future/present work and references

Conclusions

- A new version of P-Lingua has been designed including rule patterns and semantic definitions
- a command-line compiler has been written from scratch in C/C++
- a command-line simulator tool is also provided
- hard-coding the definition of the P system variants is not longer necessary
- this tool allows the designers to "play" with experimental variants of P systems

Future/present work

- To debug the code and complete the documentation
- To write simulators for parallel architectures, such as multi-core processors, pthreads, GPUs, FPGAs...
- To design optimized simulation tools for interesting case studies
- And more...

References

- I. Pérez-Hurtado et al. A new P-Lingua toolkit for agile development in membrane computing, *Information Sciences* (2022), **587**, 1–22
- M. del-Amor et al. Adaptative parallel simulators for bioinspired computing models, *Future Generation Computer Systems* (2020), **107**, 469–484
- I. Pérez-Hurtado et al. P-Lingua in two steps: flexibility and efficiency, *Journal of Membrane Computing* (2019), **1**, 93–102

Thanks for your attention!

P-Lingua