

# An environment for virtual experimentation with computational models based on P systems

PhD dissertation by  
Luis Valencia Cabrera

Supervisors  
**Mario de Jesús Pérez Jiménez**  
**Agustín Riscos Núñez**

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

February 2, 2015. Sevilla



- 1 Introduction
- 2 Core
- 3 Applications
- 4 Conclusions

- 1 Introduction
  - Motivation
  - Models in real life
  - Bio-inspired Computing
- 2 Core
- 3 Applications
- 4 Conclusions

## 1 Introduction

- Motivation
- Models in real life
- Bio-inspired Computing

## 2 Core

## 3 Applications

## 4 Conclusions



## Starting point

- There is nothing more practical than a good theory, but...
- There is nothing more enriching for a theory than putting it into practice.

## What about Membrane Computing?

- **Theory** supported by *solid foundations*.
- **Modelling** and **Simulation** tools play an important role.

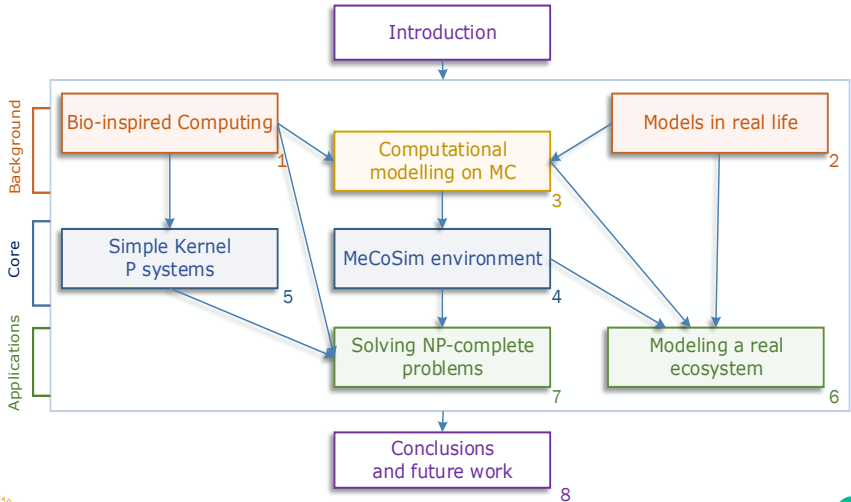
## General goals

- Providing an integrated **methodology** for the practical use of solutions based on Membrane Computing:
  - **Problem**  $\longrightarrow$  **solution**.
- Developing **software tools** to support this methodology, helping:
  - P systems designers.
  - End users.
- Applying methodology and tools to **solve relevant** theoretical and real-life **problems**.

- This work falls within **Natural Computing**.
- More specifically, **Membrane Computing**.
- Computer-aided **modelling**, **simulation** and **verification**.

- **Methodology** from *problem* to end user *solution*.
- **MeCoSim** (environment and plugins):
  - **IDE for P systems designers.**
  - **End-user visual applications.**
- **Solutions to relevant problems** (NP-hard and real-life).

# Overview - General outline



- 1 Introduction
  - Motivation
  - Models in real life
  - Bio-inspired Computing
- 2 Core
- 3 Applications
- 4 Conclusions

# The need for model designing

## Real-life complexity

- Many phenomena in real world are **complex** dynamic **systems**.
- A suitable way to analyze them is by **modelling**.

## Model

Concrete, abstract, graphical or formal representation to study, analyze, explain and reason about a system (simplified image).

- Intrinsic to any **scientific activity**.
- **Formal models**: important achievements when converging Biology, Computer Science, etc. (multidisciplinary approach).

## Formal model

**Abstraction** of a specific aspect of the world onto a **mathematical domain**.

## Features of a *good* model<sup>1</sup>

- *Relevance.*
- *Understandability.*
- *Extensibility.*
- *Mathematical-computational tractability.*

Computational modelling and simulation are in the core of modern scientific method.

---

<sup>1</sup>Regev, A., Shapiro, E. Cellular abstractions: Cells as computations. *Nature* **419**, 6905 (2002), 343-343.



## Modelling approaches

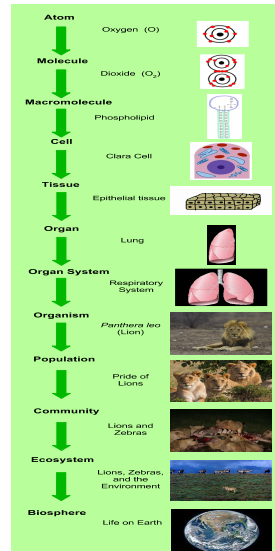
- **ODEs/PDEs.**
- **Agent based systems.**
- **Petri nets.**
- **Process algebra,  $\pi$ -calculus.**

- 1 Introduction
  - Motivation
  - Models in real life
  - Bio-inspired Computing
- 2 Core
- 3 Applications
- 4 Conclusions

# Getting inspired by Nature

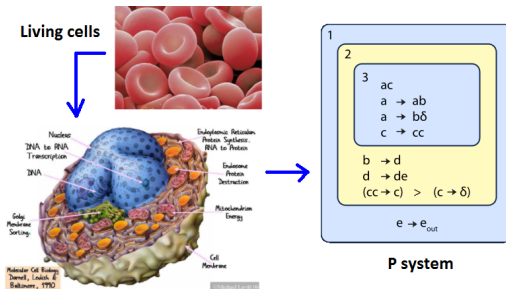
## Natural Computing

*Analysis, design and implementation of processes that can be considered as calculation procedures.*



# Membrane Computing

- Inspired by the *structure* and *functioning* of **living cells**.
- **Machine-oriented** computational paradigm.
  - Solid *theoretical foundations* (formalization and computational complexity).
  - *Solutions* to theoretical or practical problems.



## Main *classical* frameworks

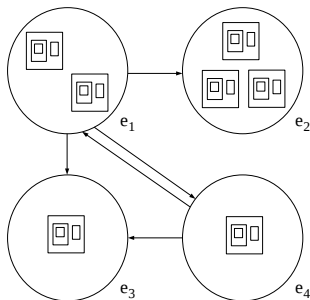
- Cell-like P systems.
- Tissue-like P systems.
- Spiking Neural P systems.

## More *recent* frameworks

- Multienvironment P systems.
- Kernel P systems.

# Multienvironment P systems I

- Environments containing P systems.
- Communication among environments.
- Computable functions associated with rules.
- Semantics: *max. parallel, non deterministic and synchronized.*



## Definition

A multienvironment P system of degree  $(q, m, n)$  ( $q, m, n \geq 1$ ) and with  $T \geq 1$  time units is a tuple

$$(G, \Gamma, \Sigma, \mu, T, \Pi_1, \dots, \Pi_n, \mathcal{R}, E_1, \dots, E_m, \mathcal{R}_E)$$

where:

- $G = (V, S)$  is a directed graph. Let  $V = \{e_1, \dots, e_m\}$ .
- $\Gamma$  and  $\Sigma$  are alphabets such that  $\Sigma \subsetneq \Gamma$ .
- $\mu$  is a rooted tree with  $q$  nodes (called membranes) injectively labelled by elements from the set  $\{1, \dots, q\} \times \{0, +, -\}$ . If the label of a membrane is  $(i, \alpha)$ , then such a membrane will be denoted as  $[ ]_i^\alpha$  and we will say that the membrane has label  $i$  and electrical charge  $\alpha$ . The root of the tree has 1 as associated label.
- $T$  is a natural number.
- For each  $k, 1 \leq k \leq n$ ,  $\Pi_k = (\Gamma, \mu, \mathcal{M}_{1,k}, \dots, \mathcal{M}_{q,k}, \mathcal{R})$  are P systems of degree  $q$  such that: (a) all of them have the same membrane structure  $\mu$ ; (b)  $\mathcal{M}_{1,k}, \dots, \mathcal{M}_{q,k}$  are finite multisets over  $\Gamma$  (initial multisets); and (c)  $\mathcal{R}$  is a finite set of rules of the form  $r \equiv u[v]_i^\alpha \xrightarrow{f_r} u'[v']_i^{\alpha'}$ , where  $u, v, u', v' \in M_f(\Gamma)$ ,  $u + v \neq \emptyset$ ,  $1 \leq i \leq q$  and  $\alpha, \alpha' \in \{0, +, -\}$ . Each rule  $r$  from the P system has an associated computable function  $f_r$  whose domain is  $\{1, \dots, T\}$ .

## Definition (cont.)

- $E_1, \dots, E_m$  are finite multisets over  $\Sigma$ .
- $\mathcal{R}_E$  is a finite set of rules of the form

$$(x)_{e_j} \xrightarrow{p_r} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}} \quad y \quad (\Pi_k)_{e_j} \xrightarrow{p_{r'}} (\Pi_k)_{e_{j'}}$$

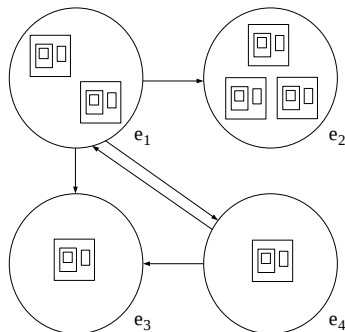
where  $x, y_1, \dots, y_h \in \Sigma$ ,  $(e_j, e_{j_l}) \in S$ ,  $1 \leq l \leq h$ ,  $(e_j, e_{j'}) \in S$ ,  $1 \leq j, j' \leq m$ ,  $1 \leq k \leq n$  and  $p_r, p_{r'}$  are computable functions whose domain is  $\{1, \dots, T\}$ . Moreover:

- If  $(x)_{e_j} \xrightarrow{p_r} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$  is a rule in  $\mathcal{R}_E$ , then there cannot exist any rule in  $\mathcal{R}$  whose left-hand side is of the form  $u[v]_1^\alpha$  with  $x \in u$ .



# Multicompartmental P systems I

- Stochastic approach.
- Initially, P systems are randomly distributed among environments.
- Computable functions: **propensities**.
- P systems and objects can be sent to other environments.
- Semantics: *multicompartmental Gillespie's algorithm*.



## Definition

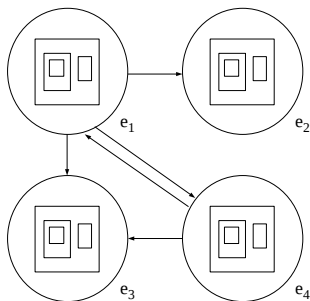
A multicompartmental P system of degree  $(q, m, n)$  ( $q, m, n \geq 1$ ) having  $T \geq 1$  time units is a multienvironment P system of degree  $(q, m, n)$  and with  $T$  time units

$$(G, \Gamma, \Sigma, \mu, T, \Pi_1, \dots, \Pi_n, \mathcal{R}, E_1, \dots, E_m, \mathcal{R}_E)$$

which fulfills the following conditions:

- The computable functions associated to the rules of the environment and the rules of the P systems are the *propensities* of such rules. These functions are determined from the values of stochastic constants, by applying the *mass action law*. The stochastic constants associated with each rule are calculated, on their turn, from the values of kinetic constants which have been experimentally calculated. The propensities are functions which depend on time, but on the other hand, they do not depend on the environment  $e_j$  where the corresponding P system currently is.
- At the initial instant, the  $n$  P systems are randomly distributed among the  $m$  environments of the system.
- For every rule of type  $(x)_{e_j} \xrightarrow{Pr} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ , we have  $h = 1$ ; that is, an object  $x$  can just move from an environment to another one, possibly getting transformed into another object  $y_1$ .

- Probabilistic approach.
- Initially, each environment holds a single P system.
- Computable functions: **probabilities**.
- Only objects can be sent to other environments.
- Semantics:
  - *DNDP*: Direct Non-Deterministic distribution with Probabilities.
  - *DCBA*: Direct distribution based on Consistent Blocks Algorithm.



## Definition

A Population Dynamics P System (**PDP system**) of degree  $(q, m)$ , where  $(q, m \geq 1)$  and having  $T \geq 1$  time units is a multienvironment P system of degree  $(q, m, m)$  and with  $T$  time units

$$(G, \Gamma, \Sigma, \mu, T, \Pi_1, \dots, \Pi_m, \mathcal{R}, E_1, \dots, E_m, \mathcal{R}_E)$$

satisfying the following conditions:

- At the initial instant, each environment  $e_j$  contains exactly one P system, which will be denoted by  $\Pi_j$ . Therefore, the number of P systems matches the number of environments.
- Functions  $p_r$  associated with rules from  $\mathcal{R}_E$  of the type

$$(x)_{e_j} \xrightarrow{p_r} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$$

have their range included in  $[0, 1]$  and they verify:

- For each  $e_j \in V$  and  $x \in \Sigma$ , the sum of the functions associated with rules of the above type is the constant function 1.
- Functions  $p_r$  associated with rules from  $\mathcal{R}_E$  of the type  $(\Pi_k)_{e_j} \xrightarrow{p_r} (\Pi_k)_{e_{j'}}$  are all constant and equal to 0; that is, one may as well assume that this type of rule is forbidden, or equivalently, P systems residing in an environment cannot travel to any other environment.
- For each rule  $r \in \mathcal{R}$  of the system  $\Pi_j$  located in  $e_j$ ,  $1 \leq j \leq m$ , the computable function  $f_r$  also depends on the environment (thus, it will be denoted as  $f_{r,j}$ ) and its range is contained within  $[0, 1]$ . Moreover, for each  $u, v \in M_r(\Gamma)$ ,  $1 \leq i \leq q$  and  $\alpha, \alpha' \in \{0, +, -\}$ , the sum of the functions  $f_{r,j}$  with  $r \equiv u[v]_i^\alpha \rightarrow u'[v']_i^{\alpha'}$ , is the constant function 1.

## Purpose

- Need of an **integrative model** with the *most relevant* **syntactic** and **semantic** ingredients of different P systems.
- Relatively low level, defined in an operational style.
- Two-fold intention:
  - **Flexible** to solve many problems from a core (*kernel*).
  - **Uniform platform** for computational modelling, simulation and verification of models based on P systems.

## Basic components

- *Compartments* in a *dynamic non directed graph*.
- *Types of compartments* defining *sets of guarded rules* and *execution strategies*.

## Definition

$T$  is a *set of compartment types*,  $T = \{t_1, \dots, t_s\}$ , where  $t_i = (R_i, \sigma_i)$ ,  $1 \leq i \leq s$ , consists of a set of rules,  $R_i$ , and an execution strategy,  $\sigma_i$ , defined over  $Lab(R_i)$ , the labels of the rules of  $R_i$ .

## Definition

A *kernel P (kP) system* of degree  $n$  is a tuple

$$k\Pi = (A, \mu, C_1, \dots, C_n, i_o),$$

where  $A$  is a finite set of elements called *objects*;  $\mu$  defines the *membrane structure*, which is a graph,  $(V, E)$ , where  $V$  are vertices indicating components, and  $E$  edges;  $C_i = (t_i, w_i)$ ,  $1 \leq i \leq n$ , is a *compartment* of the system consisting of a compartment type from  $T$  and an *initial multiset*,  $w_i$  over  $A$ ;  $i_o$  is the *output compartment* where the result is obtained.

## Definition

If  $g$  is the *abstract relational expression*  $\gamma a^n$  and the current multiset is  $w$ , then the guard denotes the *relational expression*  $\#_a(w)\gamma n$ . The guard  $g$  is true for the multiset  $w$  if  $\#_a(w)\gamma n$  is true.

## Definition

If  $g$  is the *abstract Boolean expression* and the current multiset is  $w$ , then the guard denotes the *Boolean expression* for  $w$ , obtained by replacing abstract relational expressions with relational expressions for  $w$ . The guard  $g$  is true for the multiset  $w$  when the Boolean expression for  $w$  is true.

## Definition

A guard is: (i) one of the Boolean constants *true* or *false*; (ii) an abstract relational expression; or (iii) an abstract Boolean expression.

## Definition

A rule from a compartment  $C_{l_i} = (t_{l_i}, w_{l_i})$  can have one of the following types:

- (a) **rewriting and communication** rule:  $x \rightarrow y \{g\}$ ,  
 where  $x \in A^+$  and  $y$ , has the form  $y = (a_1, t_1) \dots (a_h, t_h)$ ,  $h \geq 0$ ,  $a_j \in A$  and  $t_j$  indicates a compartment type from  $T$  – see Definition 5 – with instance compartments linked to the current compartment;  $t_j$  might indicate the type of the current compartment, i.e.,  $t_{l_i}$  – in this case it is ignored; if a link does not exist (the two compartments are not in  $E$ ) then the rule is not applied; if a target,  $t_j$ , refers to a compartment type that has more than one instance connected to  $l_i$ , then one of them will be non-deterministically chosen;
- (b) **structure changing rules**; the following types are considered:
  - (b1) **membrane division** rule:  $[x]_{t_{l_i}} \rightarrow [\gamma_1]_{t_{l_1}} \dots [\gamma_p]_{t_{l_p}} \{g\}$ ,  
 where  $x \in A^+$  and  $y_j$  has the form  $y_j = (a_{j,1}, t_{j,1}) \dots (a_{j,h_j}, t_{j,h_j})$  like in rewriting and communication rules; the compartment  $l_i$  will be replaced by  $p$  compartments; the  $j$ -th compartment, instantiated from the compartment type  $t_{l_j}$  contains the same objects as  $l_i$ , but  $x$ , which will be replaced by  $y_j$ ; all the links of  $l_i$  are inherited by each of the newly created compartments;
  - (b2) **membrane dissolution** rule:  $\square_{t_{l_i}} \rightarrow \lambda \{g\}$ ;  
 the compartment  $l_i$  will be destroyed together with its links;
  - (b3) **link creation** rule:  $[x]_{t_{l_i}}; \square_{t_{l_j}} \rightarrow [\gamma]_{t_{l_i}} - \square_{t_{l_j}} \{g\}$ ;  
 the current compartment is linked to a compartment of type  $t_{l_j}$  and  $x$  is transformed into  $\gamma$ ; if more than one instance of the compartment type  $t_{l_j}$  exists then one of them will be non-deterministically picked up;  $g$  is a guard that refers to the compartment instantiated from the compartment type  $t_{l_i}$ ;
  - (b4) **link destruction** rule:  $[x]_{t_{l_i}} - \square_{t_{l_j}} \rightarrow [\gamma]_{t_{l_i}}; \square_{t_{l_j}} \{g\}$ ;  
 is the opposite of link creation and means that the compartments are disconnected.



## 1 Introduction

## 2 Core

- MeCoSim
- Methodology
- Simple kernel P systems

## 3 Applications

## 4 Conclusions

## 1 Introduction

## 2 Core

- MeCoSim
- Methodology
- Simple kernel P systems

## 3 Applications

## 4 Conclusions

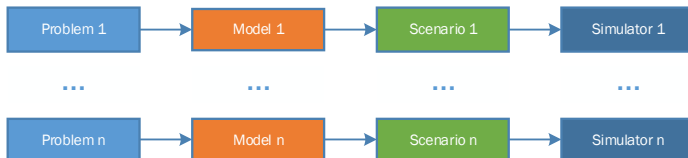
# Implementation vs simulation

- **Designs** for the **abstract machines** have been provided.
- **No implementation** (real machines).
- **Simulation** of the abstract machines on electronic devices is possible, complementing:
  - **Computational power** and **efficiency** studies.
  - **Formal verification** and **validation of properties**.
  - **Manual traces** of the abstract model computation.

# Simulation trends in Membrane Computing

## Specific purpose. A problem

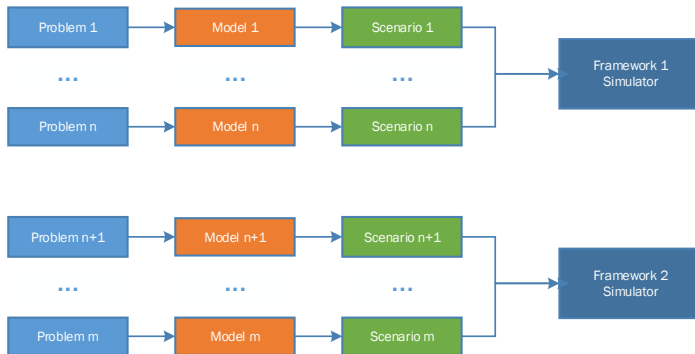
- *Ad-hoc* simulators.
- Focus: solution to a **specific problem/instance**.



# Simulation trends in Membrane Computing

## Framework oriented

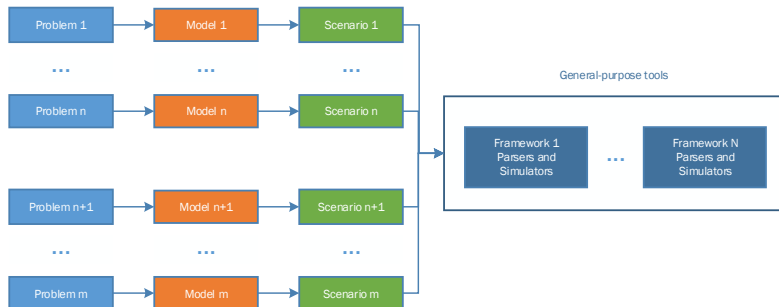
- A wider range of problems, within a **specific framework**.
- **MetaPlab, Infobiotics Workbench**, etc.



# Simulation trends in Membrane Computing

## General purpose tools

- **Global scope** (many frameworks within Membrane Computing).
- **P-Lingua framework**<sup>2</sup>.



<sup>2</sup>Developed by the Research Group on Natural Computing.

## Parsers and simulators

- Cell-like P systems.
- Tissue-like P systems.
- Spiking Neural P systems.
- **PDP systems.**
- **Simple kernel P systems.**

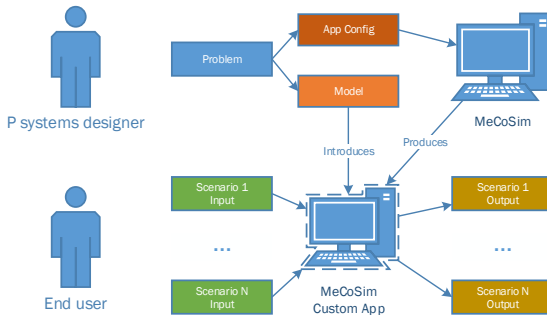
## Features to review

- High coupling instance - solution in a file.
- No high level interface for P systems designers.
- Focus on P systems designer, not on end user.

# MeCoSim (Membrane Computing Simulator)

## Goals

- For **P systems designers**: modelling, edition, testing, debugging, simulation, analysis and visualization of P systems; delivery of end-user applications.
- For **end users**: custom applications (**black boxes**), to enter inputs, run virtual experiments and get results.
- Required features: **flexibility**, **extensibility**.





## Origin

- Development of ad-hoc visual applications for different ecosystems in RGNC.
- Detection of general needs.

## Definition of a custom application

- Visual **arrangement**.
- **Input tables** to introduce data.
- **Parameters** generation for the model/solution.
- **Outputs** to show (tables/charts/graphs).

- **Modelling** and edition of solutions (P-Lingua files).
- **Debugging**.
- **Visualization** of *alphabet*, *membrane structure* and *multisets*.
- **Virtual experimentation** by **simulating** (halting or number of steps).

A **plugins architecture** has been implemented in MeCoSim.

## Plugins developed within this thesis

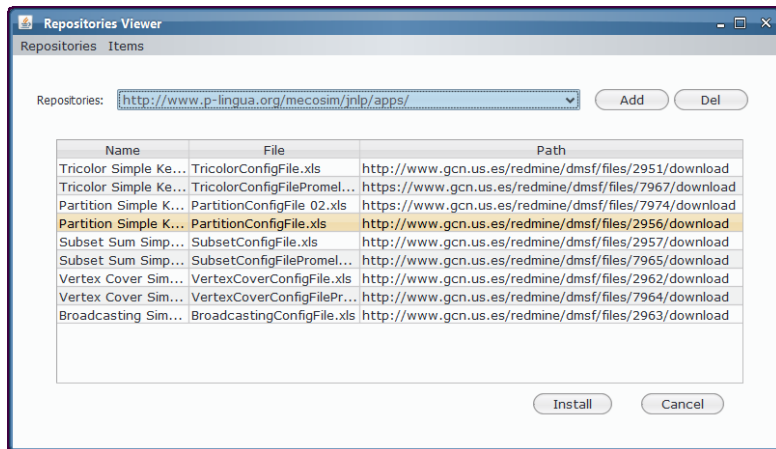
- MeCoSim basics.
- Processes.
- Graphs.
- Languages Integration.
- Daikon.
- Promela.
- SAT.

## System of repositories

- Plugins.
- Apps.
- Models.
- Scenarios.

### Example - *apps.xml*

```
<apps>
<app name="Tricolor Simple Kernel 1" config="TricolorConfigFile.xls"
      path="http://www.gcn.us.es/redmine/dmsf/files/2951/download"/>
...
</apps>
```



## 1 Introduction

## 2 Core

- MeCoSim
- **Methodology**
- Simple kernel P systems

## 3 Applications

## 4 Conclusions

## Integrated methodology

- **From:** initial study of a **problem**.
- **To:** **solution** (a custom application).
- **Through:** computational **modelling** tools based on **P systems**.
- **Supported by:** **MeCoSim**, on top of **P-Lingua**.

## Types of problems

- **Real-life** problems, complex systems as population dynamics.
- **NP-hard** problems, e.g. SAT or 3-COL.

## Main stages

- Modelling.
- Simulation.
- Customization.
- Debugging.
- Visualization and data analysis.
- Invariants detection.
- Properties verification.



Providing a **solution** or **model**, depending on the type of problem under study.

## Steps

- **Goal** and **scope**.
- **Study** of the problem.
- **Abstraction**. It may imply:
  - *Processes* and *interactions*.
  - *Parameters*.
  - *Sequencing/parallelization*.
- **Solution/model** by **P** systems.
- **Traces**.

## Purpose

- **Input:** a **model** based on P systems.
- **Result:** the **simulation** of the model.
- **Requirements:**
  - **Description** of the system.
  - **Input** data.
  - **Simulator.**

## Support

- **Description** → **P-Lingua language** and **parsers**.
- **Input** → **P-Lingua** files or **MeCoSim** custom apps.
- **Simulator** → **MeCoSim** environment + **P-Lingua** or **external** simulator.

# Simulation III

**Pandemic**

Scenario Edit Model **Simulation** Plugins Help

Simulate! Ctrl+L

Options

Number of cycles

Number of simulations (by cycle)

Simulation Algorithm

- binomial
- probabilistic
- dndp
- dndp-seq
- dndp2
- dndp2-seq
- no-maximal
- dndp3
- dndp4**
- dcba

Population Neighborhoods infect Symptoms

Community 1 Community 2 Community 3

Family	C1=X1	C2=X2	C3=X3	C4=X4
1	1	1	0	2
2	0	0	2	1
3	1	0	1	2
4	1	1	0	2
5	0	0	2	1
6	1	0	1	2
7	1	1	0	2
8	0	0	2	1
9	1	0	1	2
10	1	1	0	2
11	1	1	0	2
12	0	0	2	1
13	1	0	1	2

Probability to recover

A1=X6	A2=X7
1	
0	
0	
1	
0	
0	
1	
0	
0	
1	
0	1
0	0
1	0
0	0

**P SYSTEM USER**

**Scenario Data:** C:\Users\Muevo\Desktop\aaa\pandemia\_2.ec2

**Model:** C:\Users\Muevo\Desktop\aaa\pandemic2012.pli

Simulated cycles: 80

Simulations by cycle: 2

Steps by cycle: 18

Selected simulator: dndp4

0%

(c) 2011 Research Group on Natural Computing. <http://www.gcn.us.es>

## Purpose

- Every **problem** deals with different **data**:
  - **Input** data, describing **instances**.
  - Expected **output** *results*.
- Goal: **custom application**, providing:
  - **Black box** for *end-user* virtual experiments.
  - **Usability** for P systems designers.

## Support

- *Custom apps generation* → **MeCoSim**.
- Custom elements, as shown before:
  - **Arrangement** of inputs and outputs.
  - **Input** tables.
  - **Parameters** generation.
  - **Output** results (tables, charts, graphs).

## Purpose

- Check **syntactic correctness**.
- Analyze the **dynamics** of the system:
  - Observing the **evolution** of the system.
  - Testing **matching** with **expected** behavior.

## Errors detection (*iterative process*)

- Theoretical model.
- Computational description.
- Simulators.

## Purpose

- Analysis of **P systems** for designers.
- **Data analysis** of parts of the system output, post-processing.
  - *Aggregation.*
  - *Basic descriptive statistics.*
  - *Charting* to analyze evolution or distribution of elements.

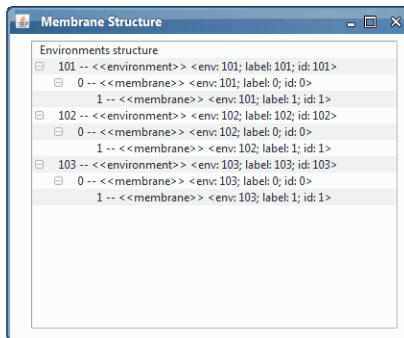
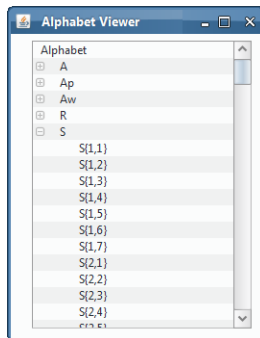


## Support

- **Viewers** for structures and elements of P systems configurations.
- **Custom** outputs. **Post-processed, filtered** and **aggregated outputs**.
- **Charts** generation.
- **Graphs**.

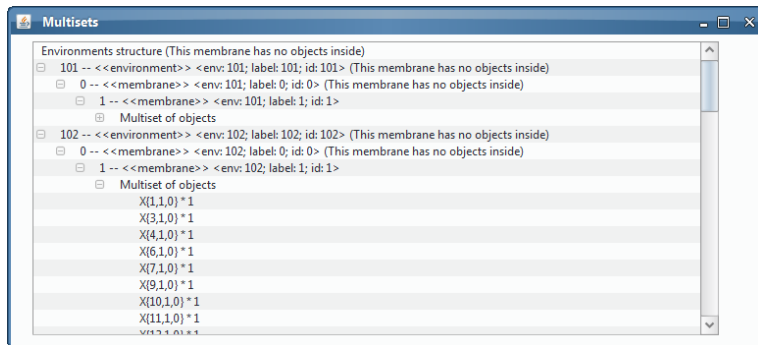
# Visualization and data analysis III

## Advanced visualization of structures I



# Visualization and data analysis IV

## Advanced visualization of structures II



# Visualization and data analysis V

## Graphs visualization I

### Graphs

A plugin was developed to generate graphs from parameters, oriented to:

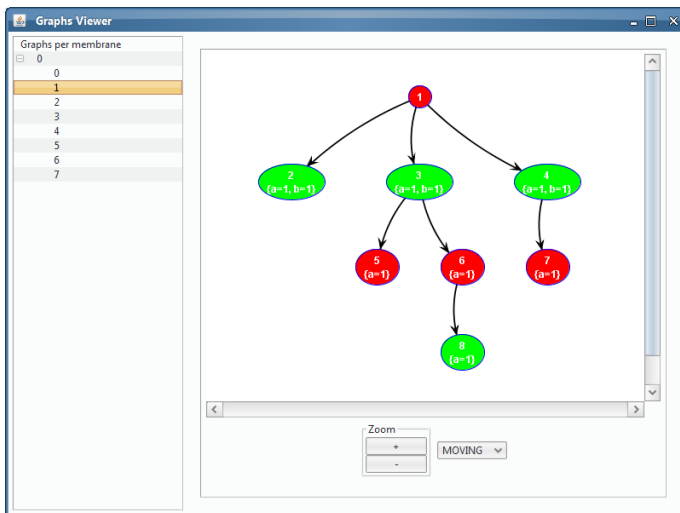
- **Designers**: P system inner **structure**.
- **End users**: Information coded in the **solution** to the problem itself.

Options for:

- **Simple** graph or tree.
- **Tree view** of a set of graphs or trees.

# Visualization and data analysis VI

## Graphs visualization II



## Purpose

- Solutions and models may present *interesting properties*.
- Not easy to detect manually.
- Goal: **automatic invariants detection**.

## Support

- MeCoSim **custom outputs** focusing specific data.
- MeCoSim **Daikon plugin**<sup>3</sup> configuration for extracting *traces* from outputs.
- **Daikon invariants detector**<sup>4</sup> analyzing traces to detect *invariants*.

---

<sup>3</sup>Daikon and Promela plugins developed in collaboration with the Department of Mathematics and Computer Science, University of Pitesti, and the Department of Computer Science, University of Sheffield

<sup>4</sup>Daikon: implementation of dynamic detection of likely invariants, developed by the Programming Languages and Software Engineering of the University of Washington

## Purpose

- Input:
  - A **design of a P system** based model/solution.
  - Syntactically and semantically **correct**.
  - (Opt.) Previous manual definition and verification of properties.
- Goal: **automatic verification of properties**. May use model checking techniques.



## Support

- MeCoSim integration with tools for formal verification:
  - **Promela**<sup>5</sup> code generation **from MeCoSim**.
  - **Properties definition** can be added to generated Promela file.
  - **Spin**<sup>6</sup> model checker runs from Promela code.

---

<sup>5</sup>**Promela** (Process Meta Language): modelling language accepted by Spin model checker.

<sup>6</sup>**Spin** model checker: software for automatic formal verification of multi-threaded applications, developed at Bell Labs in the Unix group of the Computing Sciences Research Center, starting in 1980.

# Properties verification III

The screenshot shows the Promela model checker interface with the file `tricolor_kernel.pli` open. The interface is divided into three main panes:

- P-Link:** Contains the Promela source code. The code defines a main function that calls `three_colouring(n)`, which in turn calls `init_cells()` and `init_rules(n)`. The `init_rules(n)` function contains several guarded rules for a tricolor kernel, including rules for setting initial values and maintaining invariants.
- Promela:** Shows the generated Spin input file. It includes mandatory boundary constants, model-specific constants, and the alphabet of symbols used in the model. The code is commented with explanations of the constants and the mapping of symbols to integers.
- Spin:** Displays the standard output of the command. It shows the execution state of the p system, including the current step number and the values of the variables `X`, `A`, and `B` for each cell. The output is formatted as a table with columns for the step number, the variable name, and the value.

Navigation buttons (back, forward, search) are visible between the panes. The Spin pane shows the standard output of the command, indicating the execution state of the p system.

## 1 Introduction

## 2 Core

- MeCoSim
- Methodology
- Simple kernel P systems

## 3 Applications

## 4 Conclusions

## Simple kernel P systems

Simplified version of Kernel P systems

- Only **rewriting-communication** and **division** rules.
- Only **maximal parallelism** as execution strategy.

# Simple kernel P systems

## Formalization

### Definition

A simple kernel P system of degree  $n \geq 1$  is a tuple

$$\Pi = (\Gamma, \mathcal{E}, T, G, C_1, \dots, C_n, i_{out})$$

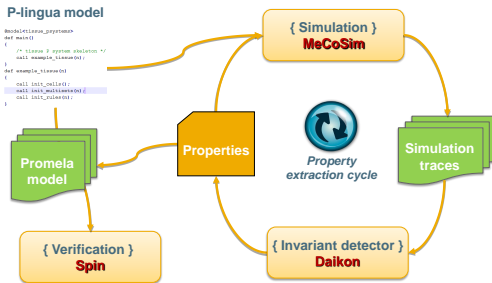
where

- $\Gamma$  is a finite alphabet whose elements are called objects.
- $\mathcal{E}$  is an alphabet contained in  $\Gamma$  (called output alphabet).
- $T = \{t_1, \dots, t_s\}$  is a set of types of compartment.
- $G = (V, E)$  is a non directed graph.
- $V = \{C_1, \dots, C_n\}$  is the set of nodes in the graph, called compartments of the system, such a way that  $C_i = (t_i, w_i)$ ,  $1 \leq i \leq n$ , with  $t_i \in T$  and  $w_i \in M_f(\Gamma)$ .
- $i_{out} \in \{0, 1, \dots, n\}$  is a number representing the environment (if  $i_{out} = 0$ ) or a compartment of the system, if  $i_{out} \in \{1, \dots, n\}$ .

# Software tools for simple kernel P systems

## Tools developed within this thesis

- A framework for modelling, simulation and verification.<sup>7</sup>
- An extension of P-Lingua language and parser for simple kernel P systems.
- Simulators: (1) In P-Lingua; (2) External (Spin)<sup>3</sup>, integrated with MeCoSim.



<sup>7</sup>In collaboration with the Department of Mathematics and Computer Science, University of Pitesti, and the Department of Computer Science, University of Sheffield

# A language for simple kernel P systems I

- **Guards:**

A rule  $a \rightarrow b \{ = a^2 \}$  is defined as:

@guard  $\{ = +a*2 \} \ ? \ [a \dashrightarrow b] ;$

- **Initialization of compartments:**

@mu (0) = [ c\*3 ]'1;

- **Definition of initial multisets:**

Associated with each specific compartment:

@mu = [ [ a\*2 ]'1 [ b ]'2 [ c\*2 ]'2 ]'0;

Associated with all the compartments of a certain type:

@ms (1) = x;

@ms (2) = y\*3;

- **New rule types in P-Lingua:**

- **Rewriting and communication rules:**

@guard  $g$  ?  $[a]'t_0 \rightarrow [a_1]'t_1, \dots, [a_h]'t_h$

- **Division rules:**

@guard  $g$  ?  $[a]'t \mid \rightarrow [v_1]'t_{i_1}, \dots, [v_p]'t_{i_p};$

Both types of rules admit internal iterators:

$[a]'1 \mid \rightarrow [b]'2 \ \& \ \{ [c, d\{i\}]' \{i\} \} : \{ 3 \leq i \leq n \};$

- **Internal iterators:**

- ① **Over multisets:**  $\& \{ multiset \} : \{ index\_ranges \}$

- ② **Over compartments:**  $\& \{ [ multiset ]' \{ label \} \} : \{ index\_ranges \}$

- ③ **Over guards:**

$\& \{ guard \} : \{ index\_ranges \}$ , for conjunction, and

$\mid \{ guard \} : \{ index\_ranges \}$ , for disjunction.



## 1 Introduction

## 2 Core

## 3 Applications

- Solving NP-complete problems
- Modelling of a real ecosystem

## 4 Conclusions

- 1 Introduction
- 2 Core
- 3 Applications
  - Solving NP-complete problems
  - Modelling of a real ecosystem
- 4 Conclusions

# Solving NP-complete problems

- Membrane Computing solve **NP-hard** problems.
- This power can **not** be **fully exploited** yet.
- Tools as MeCoSim **can aid studies** in:
  - Tasks of design, debug, simulation and traces analysis, verification, etc.
  - Practical use to solve relevant instances.

## Problems studied

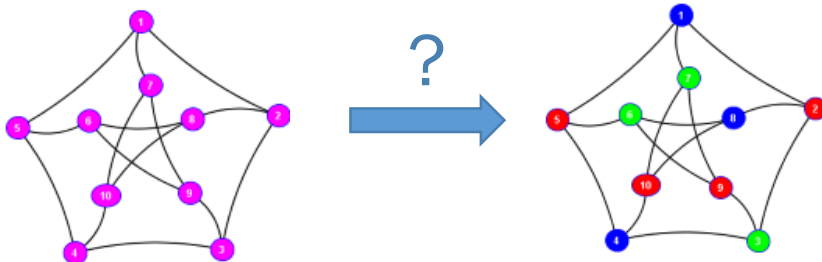
- Cell-like P systems with active membranes: SAT, Partition, Subset Sum, Knapsack.
- **Tissue-like P systems with cell division:** 3-COL (colorings), SAT.
- Tissue-like P systems with cell separation: SAT.
- Spiking-neural P systems: SAT.
- **Simple kernel P systems:** 3-COL, Partition, Subset sum, Vertex cover.

# A solution for 3-COL problem based on simple kernel P systems I

## 3-COL problem

***Given a non-directed graph, determine if a valid 3-coloring exists.***

- Solve many practical problems (mainly in communication networks).



## Purpose

- **Simple kernel P system** solving the decision problem (yes/no).
- Solution based on a brute force algorithm:
  - 1 **Generation of colorings**, based on division rules.
  - 2 **Checking**, controlled by guards.
  - 3 **Output**, sending *yes/no* to environment.

# A solution for 3-COL problem based on simple kernel P systems III

## Solution

A family of *simple kernel P systems* is considered:

$\Pi = \{\Pi(n) : n \in \mathbb{N}\}$ , where  $\Pi(n) = (\Gamma, IO, T, G, C_1, C_2, i_{out}, i_{in})$  is defined as follows:

- The working alphabet  $\Gamma$  is the following set:

$$\begin{aligned}\Gamma = & \{A_1, \dots, A_n\} \cup \{A_{i,j} \mid 1 \leq i < j \leq n\} \cup \{T_1, \dots, T_n\} \cup \\ & \{B_1, \dots, B_n\} \cup \{R_1, \dots, R_n\} \cup \{G_1, \dots, G_n\} \cup \\ & \{a, s, X, Y, Z, \text{yes}, \text{no}\} \cup \{X_1, \dots, X_{2n+3}\}\end{aligned}$$

Where:

- $A_i$ ,  $1 \leq i \leq n$ , are symbols representing the  $n$  vertices in  $\mathcal{G}$ .
- $A_{i,j}$ ,  $1 \leq i < j \leq n$ , are symbols representing the possible edges in  $\mathcal{G}$ .
- $T_i$ ,  $1 \leq i \leq n$ , are symbols used in the process dividing compartments of type  $C_2$ .
- $B_i, R_i, G_i$ ,  $1 \leq i \leq n$ , are symbols codifying the three colors (as in the case of tissue-like P systems).
- Symbol  $a$  is used only in compartment  $C_1$  to select a single answer to be later sent to the environment.
- $s, X, Y$  are symbols user in compartments of type  $C_2$ .
- $Z$  is a symbol to be sent to compartment  $C_1$ .
- $\text{yes}, \text{no}$  are the possible answers: one of them will reach the environment from  $C_1$  in the last step of the computation, as required by the decision problem.
- Symbols  $X_1, \dots, X_{2n+3}$  are used as a counter to control the maximum number of steps  $(2n+2)$  required by the only possible input from compartments of type  $C_2$ .

## Solution (cont.)

- $IO = \{\text{yes}, \text{no}\}$ .
- $T = \{t_1, t_2\}$ , with  $t_1 = (R_1, \sigma_1)$  and  $t_2 = (R_2, \sigma_2)$ ; that is, there exist two types of compartments  $t_1$  and  $t_2$ . The execution strategy  $\sigma_1 = \sigma_2$  is the maximal parallelism with the constraint for each compartment to apply, at most, a division rule for each step of the computation.

The sets of rules are the following:

- $R_1$  is the set of rules:
  - ★  $r_{1,i} : X_i \rightarrow X_{i+1}, 1 \leq i \leq 2n+2$ .
  - ★  $r_{1,2n+3} : aZ \rightarrow (\text{yes}, 0)$ .
  - ★  $r_{1,2n+4} : aX_{2n+3} \rightarrow (\text{no}, 0) \{ \geq \bar{Z} \}$ .

Rules  $r_{1,i}, 1 \leq i \leq 2n+2$  are responsible for counting the first  $2n+2$  steps; during that stage, for each solution found, an object  $Z$  is sent from  $C_2$  to  $C_1$ ; if one or more  $Z$  objects were received from compartments of type  $C_2$ , that is, there exist at least one solution, then the compartment of type  $C_1$  sends yes to the environment; otherwise, if no  $Z$  object has been received, after  $2n+3$  steps an object  $no$  is sent;



## Solution (cont.)

- $R_2$  is the following set of rules:

*Division rules:*

- ★  $r_{2,2i-1} : [A_i]_2 \rightarrow [R_i A_{i+1}]_2 [T_i]_2 \{= s\}.$
- ★  $r_{2,2i} : [T_i]_2 \rightarrow [B_i A_{i+1}]_2 [G_i A_{i+1}]_2, 1 \leq i \leq n-1.$
- ★  $r_{2,2n-1} : [A_n]_2 \rightarrow [R_n X]_2 [T_n]_2 \{= s\}.$
- ★  $r_{2,2n} : [T_n]_2 \rightarrow [B_n X]_2 [G_n X]_2.$

These rules are applied at most in  $2n$  steps, where all possible colorings are obtained for the  $n$  vertices.

*Rewriting and communication rules:*

- ★  $r_{2,2n+1} : s \rightarrow \lambda \{g\}$  with  $g$  the following guard

$$\begin{aligned}
 &= A_{1,2} = B_1 = B_2 \vee = A_{1,2} = G_1 = G_2 \vee = A_{1,2} = R_1 = R_2 \vee \\
 &\quad \vee \dots \vee \\
 &= A_{n-1,n} = B_{n-1} = B_n \vee = A_{n-1,n} = G_{n-1} = G_n \vee = A_{n-1,n} = R_{n-1} = R_n
 \end{aligned}$$

- ★  $r_{2,2n+2} : X \rightarrow Y.$
- ★  $r_{2,2n+3} : Ys \rightarrow (Z, 1).$

## Solution (cont.)

The guard appearing in rule  $r_{2,2n+1}$  contains  $3n(n-1)/2$  terms and checks, for each pair  $1 \leq i < j \leq n$ , if the colors in nodes  $i$  and  $j$  is the same. If the current compartment contains object  $s$ , then that object is removed and, from that moment, no rule will be applied to that compartment.

Rule  $r_{2,2n+2}$  will transform  $X$  into  $Y$  once all possible calculation have finished.

Rule  $r_{2,2n+3}$  will be applied a solution is present in the current compartment of type  $C_2$ , and will send object  $Z$  to the compartment with type  $C_1$ .

- $G = (\{1, 2\}, \{\{1, 2\}\})$ .
- $C_1 = (t_1, w_1)$ ,  $C_2 = (t_2, w_2)$ , where  $w_1 = aX_1$ ,  $w_2 = A_1 s$ .
- $i_{out} = 0$ ; that is, the output of the system is codified in the environment.
- $i_{in} = 2$ ; that is, the input compartment is the initial  $C_2$ .

It is important to note that instance  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  of 3-COL problem will be processed by the *simple kernel P system*  $\Pi(s(\mathcal{G}))$  with input multiset  $cod(\mathcal{G})$ .

## Instances of 3-COL

Following two computable functions,  $s$  and  $cod$ , are defined over the set of instances of 3-COL problem as follows:

$$\begin{cases} s(\mathcal{G}) &= |\mathcal{V}| \\ cod(\mathcal{G}) &= \{A_{i,j} : \{i,j\} \in \mathcal{E}, 1 \leq i < j \leq n\} \end{cases}$$

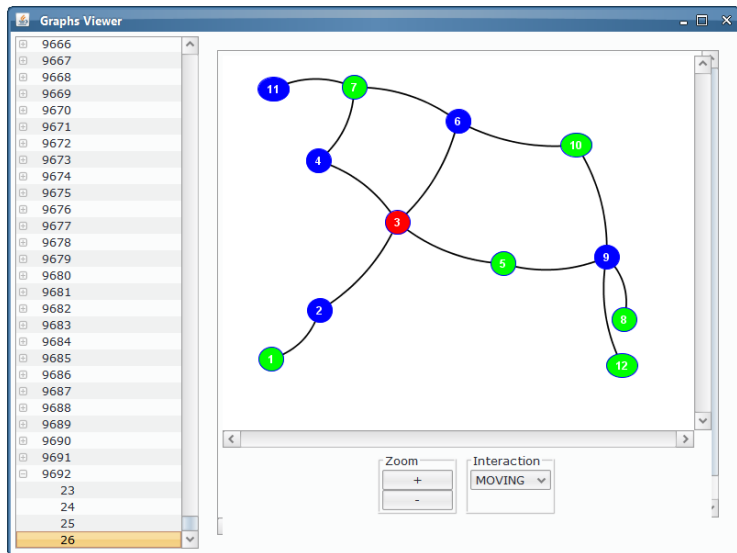
with  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  a non directed graph containing at least two vertices.

- P-Lingua file.
- Input tables for nodes and edges.
- Outputs for decision (yes/no) and colorings.
- Parameters for:
  - $n$  (number of nodes),  $m$  (number of edges).
  - $e_{i,1}$  and  $e_{i,2}$  (the two nodes of each edge  $i$ ).
  - Parameters for generating coloring information.

### 3-COL. Parameters

Param Name	Param Value	Index 1	Index 2
n	<2,1,1>		
ne	<2,1,2>		
e	<4,\$1\$,\$2\$>	[1..ne]	[1..2]
nm	<@r,7>		
m	<7,\$1\$,1>	[1..nm]	
g	m{\$2\$}	[1..@steps]	[1..nm]
m	ne		
versInfo	<,6,\$1\$,\$2\$>	[1..<@r,6>]	[1..3]
versInfo2	<,6,\$1\$,3-\$2\$>	[1..<@r,6>]	[1..2]
versInfo2	<,6,\$1\$,\$2\$>	[1..<@r,6>]	3

## 3-COL. Tree view of graphs



Focus: **generation** phase of 3-COL (confluence).

```
@model<tissue_psystems>
def main()
{
    call init_cells();
    call init_multisets(n);
    call init_rules(n);
}
def init_cells()
{
    @mu = [[]'2]'0;
}
def init_multisets(n)
{
    @ms(2) += A{i} : 1<=i<=n;
}
def init_rules(n)
{
    /* r1 */ [A{i}]'2 --> [R{i}]'2 [T{i}]'2 : 1<=i<=n;
    /* r2 */ [T{i}]'2 --> [B{i}]'2 [G{i}]'2 : 1<=i<=n;
}
```

# Generation of colorings in MeCoSim II

- Solution: **a family** of P systems.
- Input: parameter  $n$ .
- Output: **number of cells** per step.
- **Custom app**.



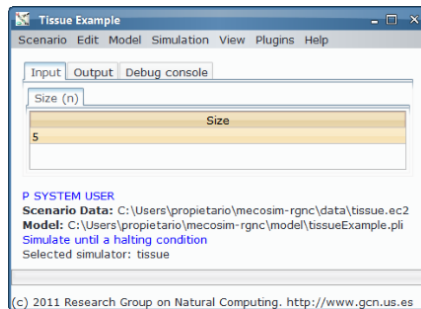
# Generation of colorings in MeCoSim III

Tab Id	Tab Name	Tab Parent Id
1	Tissue Example	0
2	Input	1
3	Size (n)	2

Table Id	Table Name	Tab Id	Columns	Init Rows	Save To File	Input / Output
1	Size (n)	3	1	1	TRUE	Input

Column Id	Column Name	Default Value	Editable	Tooltip	GraphicRole
1	Size	3	TRUE	n	

Param Name	Param Value
n	<1,1,1>



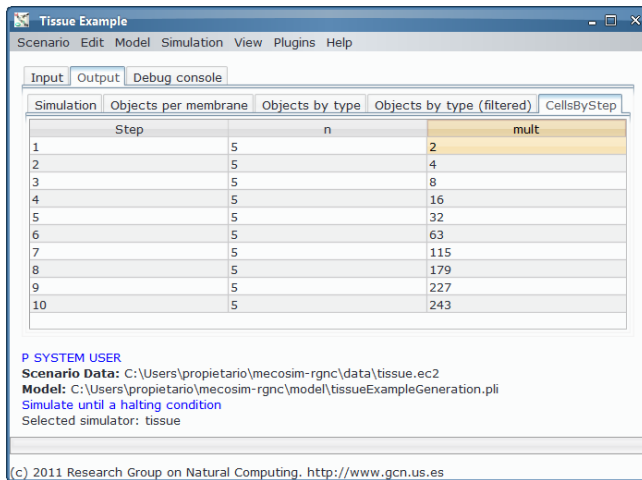
# Generation of colorings in MeCoSim IV

Result Table Id	Result Table Name	Table Id	Referred Table Id
1	CellsByStepAux	0	0
2	CellsByStep	2	1

Criteria Id	Select/Where/Group	Criteria	Formula	ReferredCriteria Id	Argument	Qualified Name	Where condition type	Where condition
1	Select	step						
2	Select	membraneID						
3	WhereAux	membraneID					Integer	0
4	Where	formula	NOT	3				
5	Group	step						
6	Group	membraneID						

Criteria Id	Select/Where/Group	Criteria	Formula	ReferredCriteria Id	Argument	Qualified Name	Where/Select condition type	Where condition
1	Select	step						
2	Auxiliary	parameter	n				Integer	
3	Select	formula	CONVERT	2	INT	valn	DirectString	
4	Auxiliary	membraneID						
5	Select	formula	COUNT	4		ncells		
6	Group	step						

# Generation of colorings in MeCoSim V



The screenshot shows the 'Tissue Example' window in MeCoSim V. The 'Output' tab is selected, displaying a table with simulation results. The table has three columns: 'Step', 'n', and 'mult'. The 'mult' column is highlighted in yellow. Below the table, the 'P SYSTEM USER' section provides details about the scenario data, model, simulation condition, and selected simulator. The footer indicates the copyright information for the Research Group on Natural Computing.

Simulation	Objects per membrane	Objects by type	Objects by type (filtered)	CellsByStep
	Step	n		mult
1	5			2
2	5			4
3	5			8
4	5			16
5	5			32
6	5			63
7	5			115
8	5			179
9	5			227
10	5			243

**P SYSTEM USER**  
**Scenario Data:** C:\Users\propietario\mecsim-rgnc\data\tissue.ec2  
**Model:** C:\Users\propietario\mecsim-rgnc\model\tissueExampleGeneration.pli  
**Simulate until a halting condition**  
Selected simulator: tissue

(c) 2011 Research Group on Natural Computing. <http://www.gcn.us.es>

# Generation of colorings in MeCoSim VI

General Daikon Data			ExtractionId	ExtractionType	ObjectName	ExtractionDataIndex
StepIndex	Enabled		1	Named	step	1
1	TRUE		2	Named	n	2
			3	Named	numcells	3

The screenshot shows the Daikon Utility window with the following content:

**File**

**Input**  
P system trace received from MeCoSim

```
step int n int numcells int
step int n int numcells int
0      0      0
1      5      2
1      5      2
2      5      4
2      5      4
3      5      8
3      5      8
4      5      16
4      5      16
5      5      32
```

**Transformed trace for Daikon**

```
DECLARE
Function::ENTER
step
int
int
1
n
int
int
```

**Output**  
Invariants produced by Daikon

```
Daikon version 4.2.9, released September 1, 2006; http://pag
Reading declaration files
[21:07:18]: Reading Tissue Example-sim1-CellsByStep-for-d
=====
Function::ENTER
step == step
n == n
numcells == numcells
0 <= step <= 63
step <= 9
step >= 0
step >= 0
n one of { 0, 5 }
0 <= n <= 63
n <= 5
n == 0 (mod 5)
n >= 0
n >= 0
numcells <= 227
numcells >= 0
numcells >= 0
```

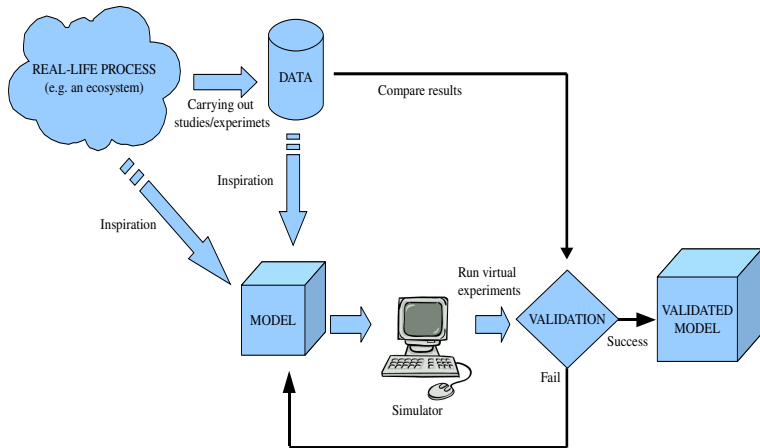
# Generation of colorings in MeCoSim VII

Extraction	Result	Valid
<b>No. cells per step</b> , $0 \dots n$	$numcells = 2^{step}$	<b>true</b>
No. cells per step $(n+1) \dots 2n$	$numcells = 3 \pmod{4}$	<b>false</b>
No. cells per step $(n+1) \dots (n + (n/2) + 1)$	$numcells = 3 \pmod{12}$	<b>false</b>
<b>No. cells last step</b> , for different values of $n$ put together	$numcells = 3^n$	<b>true</b>

- 1 Introduction
- 2 Core
- 3 Applications
  - Solving NP-complete problems
  - Modelling of a real ecosystem
- 4 Conclusions

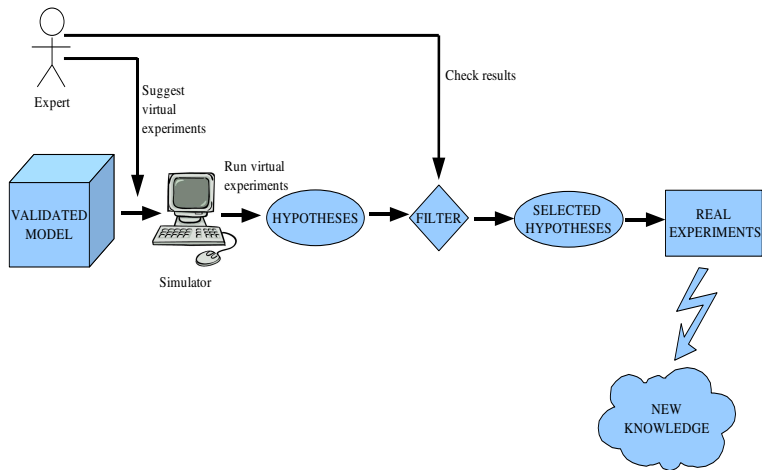
# Modelling protocol

## Experimental validation



# Modelling protocol

## Virtual experimentation





## Basic needs of ecologists

- Input:
  - Specific values for the parameters of the model.
  - Initial populations for each proposed scenario.
  - Environmental conditions.
- General mechanism for setting simulation parameters.
- Outputs generation and visualization.

## Endangered species

- Bearded vulture (Catalan Pyrenees, *Spain*).
- Avian scavengers (Catalan Pyrenees, *Spain*).
- Scavengers birds (Catalan and Navarre Pyrenees, *Spain*, and Swaziland, *South Africa*).
- Newt (*Calotriton asper*, Sierra del Cadí, *Spain*).
- Amur Tiger (*China*).
- Giant Panda (captivity, Chengdu, *China*).

## Others

- Hazel Grouse (reintroduction in the Pyrenees, *Spain*).
- Plant communities (Alt Pallars-Aran, Cerdanya-Alt Urgell, Cadí and Freser-Setcases, alpine and subalpine areas of Pyrenees, *Spain*).
- Pyrenean Chamois (pestivirus, Catalan Pyrenees, *Spain*).
- Amphibians (growth by random changes, Pi Valley stream, Sierra del Cadí, *Spain*).
- Pigs (animal production, *Spain*).
- **Zebra Mussel** (invasive species, reservoir of Ribarroja, *Spain*).
- Tritrophic interactions.
- Pandemics.

# A real ecosystem: Zebra Mussel in Ribarroja

## Context

- **Zebra Mussel** (*Dreissena polymorpha*): exotic invasive species.
- Important environmental and economical **damage**.
- In Spain since **2001**, **reservoir of Ribarroja**, managed by **Endesa S. A.**

## Purpose

- **Knowledge** about *species+environment*.
- **Hypothesis** about its **introduction**.
- Management tool to aid in **decision making** process:
  - By: **managers** designing strategies.
  - To: **eradicate or decrease** the population.

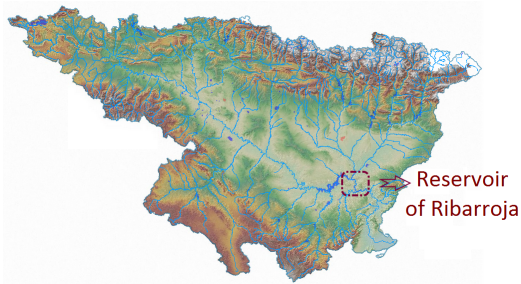
# Zebra Mussel. Biology of the species

- Aquatic bivalve mollusc, dioecious species 1:1.
- **High reproductive potential:**
  - Depending on **age** (tens to hundreds of thousands).
  - Reproductive cycle depending on the temperature.
- **High mortality rate** from spawning to juvenile (up to 98%).
- **Fast life cycle** with two stages:
  - **Larval** phase (in water column, planktonic state).
  - **Settled** phase (**young** and **adult**, benthonic state).
- Huge **mobility** and great **dispersal ability**.
- **Engineers** of the ecosystem.

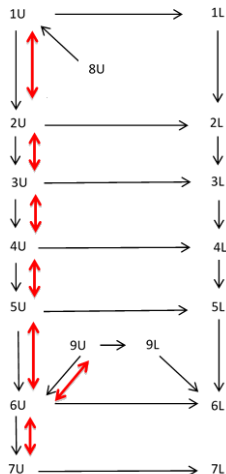
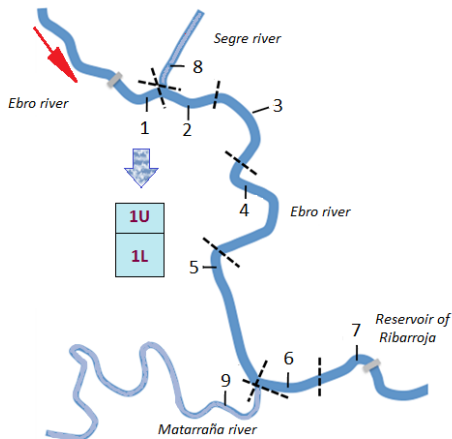


# The reservoir of Ribarroja

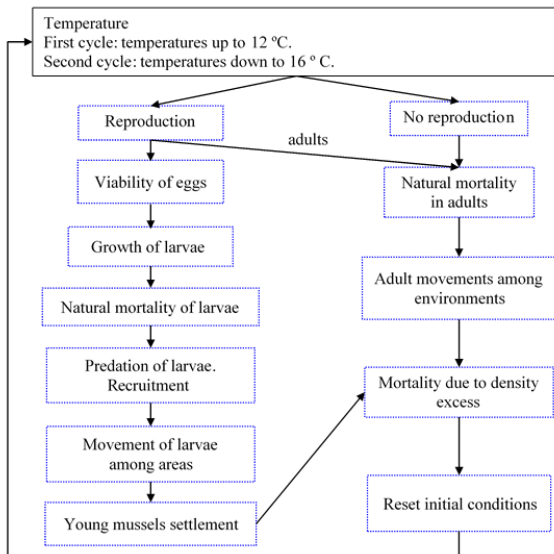
- Ebro River basin, Northeast of Spain.
- 35km length, variable depth up to 28m.
- Water from: rivers Segre and Matarraña, reservoir of Mequinenza.
- Regions with different temperature, conditions and substrate.
- Human interaction: managed by Endesa S. A., water flows, traffic of vessels.



# Areas inside the reservoir



# Processes and sequencing





## Main aspects

- **PDP system** with 18 environments (areas of the reservoir).
- P systems skeleton structure:  $\mu = [ [ ]_1 \dots [ ]_{39} ]_0$ .
  - 1-20: first reproductive cycle.
  - 21-36: second reproductive cycle.
  - 37-38: Mortality depending on density of mussels.
  - 39: recruitment of larvae, depending on the population size.
- Rules influenced by many parameters.

## Parameters

- **Biology** of the species (release of eggs, mortality rates, etc.).
- **Environmental conditions:**
  - *General* (temperature to start each reproductive cycle, proportion of eggs release per week, etc.).
  - *Per area* (temperature of the water per week, probability of reproduction per week, etc.).
- Properties of the **area** (dimensions, percentage of soil of each type, etc.).
- **Human** intervention (external inoculation, probability rates to move due to traffic of vessels, water renewal, etc.).

## Rules

$$r_{e_{1,j}} \equiv (l_1 \rightarrow l^{LE_{j,1}} l')_{e_j}, 1 \leq j \leq 18$$

$$r_{e_{2,j}} \equiv (l_2 \rightarrow l^{LE_{j,2}})_{e_j}, 1 \leq j \leq 18$$

$LE_{j,c}$ : ext. inoculation of larvae in compt.  $j$ , reprod. cycle  $c$ .

$l_c$ : aux. obj. to enable inoculation, reprod. cycle  $c$ .

# Temperatures simulation to start reproductive cycle

## Rules

$$r_{e_{3,j,w}} \equiv \left( T_w \xrightarrow{p_{w+1,j}} \gamma_{w+1} T_{w+1} \right)_{e_j} \left\{ \begin{array}{l} 1 \leq j \leq 18 \\ 0 \leq w \leq 34, w \neq 19 \end{array} \right.$$

$$r_{e_{4,j,d}} \equiv \left( T_w \xrightarrow{1-p_{w+1,j}} T_{w+1} \right)_{e_j} \left\{ \begin{array}{l} 1 \leq j \leq 18 \\ 0 \leq w \leq 34, w \neq 19 \end{array} \right.$$

$p_{w,j}$ : prob. enable reprod. compt.  $j$  week  $w$ . (Affected by prob. for  $IT_1$  for weeks 1-20 and  $IT_2$  for weeks 21-36, normal prob. func.  $N(T_{w,j}, 0.5)$ ).

$T_d$ : aux. obj. controlling week.

$\gamma_w$ : aux. obj. representing favorable conditions of temperature to start reprod. cycle week  $w$ .

## Rules

$$r_{e_{5,s,j,j'}} \equiv (V_s)_{e_j} \xrightarrow{PA_{j,j'}} (V'_s)_{e_{j'}} \left\{ \begin{array}{l} 1 \leq s \leq 6 \\ 1 \leq j \leq 18 \\ 1 \leq j' \leq 18, j' \neq j \end{array} \right.$$
$$r_{e_{6,s,j}} \equiv (V_s)_{e_j} \xrightarrow{PA_{j,j}} (V'_s)_{e_j} \left\{ \begin{array}{l} 1 \leq s \leq 6 \\ 1 \leq j \leq 18 \end{array} \right.$$

$PA_{j,j'}$ : probab. adult mussel transport from compt.  $j$  to  $j'$  due to the traffic of vessels.

## Rules

$$r_{27,m} \equiv [O_m \xrightarrow{g_2} \#]_0^0, 1 \leq m \leq 36$$

$$r_{28,m} \equiv [O_m \xrightarrow{(1-g_2) \cdot (1-mo_1)} L_{1m}]_0^0, 1 \leq m \leq 20$$

$$r_{29,m} \equiv [O_m \xrightarrow{(1-g_2) \cdot mo_1} \#]_0^0, 1 \leq m \leq 20$$

$$r_{30,m} \equiv [O_m \xrightarrow{(1-g_2) \cdot (1-mo_2)} L_{1m}]_0^0, 21 \leq m \leq 36$$

$$r_{31,m} \equiv [O_m \xrightarrow{(1-g_2) \cdot mo_2} \#]_0^0, 21 \leq m \leq 36$$

$g_2$ : % eggs loss (by filtering and predation)

$mo_c$ : mortality of eggs in reproductive cycle  $c$  (not having fertilized).

$O_m$ : eggs produced during week  $m$  of reproductive cycle.

# Movement by hydraulic regime

## Rules

$$r_{e_{13,j,m,i,j_1,j_2}} \equiv (L_{m,i,j,j_1} \xrightarrow{PR_{m,j_1,j_2}} L_{m+1,i+1,j,j_2})_{e_j} \left\{ \begin{array}{l} 1 \leq j \leq 18 \\ 1 \leq m \leq 42 \\ 0 \leq i \leq 3 \\ 1 \leq j_1 \leq 7 \\ 1 \leq j_2 \leq 7 \end{array} \right.$$

$$r_{e_{14,j,m,i,j_1}} \equiv (L_{m,i,j,j_1} \xrightarrow{PR_{m,j_1,8}} \#)_{e_j} \left\{ \begin{array}{l} 1 \leq j \leq 18 \\ 1 \leq m \leq 42 \\ 0 \leq i \leq 3 \\ 8 \leq j_1 \leq 14 \end{array} \right.$$

$$r_{e_{27,m,i}} \equiv (L_{m,i,18,16} \xrightarrow{0.5} L_{m+1,i+1,18,16})_{e_{18}} \left\{ \begin{array}{l} 1 \leq m \leq 42 \\ 0 \leq i \leq 3 \end{array} \right.$$

$$r_{e_{28,m,i}} \equiv (L_{m,i,18,16} \xrightarrow{0.5} L_{m+1,i+1,18,6})_{e_{18}} \left\{ \begin{array}{l} 1 \leq m \leq 42 \\ 0 \leq i \leq 3 \end{array} \right.$$

$PR_{m,j,j'}$ : prob. larvae mov. week  $m$ ,  
from compt.  $j$  to  $j'$ .  
 $L_{m,i,j,j'}$ : larvae released week  $m$   
having spent  $i + 1$  weeks in water  
column, and moved from  $j$  to  $j'$ .

## Rules

$$r_{e_{34,j,i}} \equiv (\rho_i \rightarrow A_i^{\varphi_j} AR^{2500})_{e_j} \left\{ \begin{array}{l} 1 \leq j \leq 18 \\ 1 \leq i \leq 2 \end{array} \right.$$

$\varphi_j$ : capacity compt.  $j$  (dep. on types of soil present).

$\rho_i$ ,  $A_i$ ,  $AR$ : aux. objs. to synchronize carrying capacity and recruitment of young individuals.



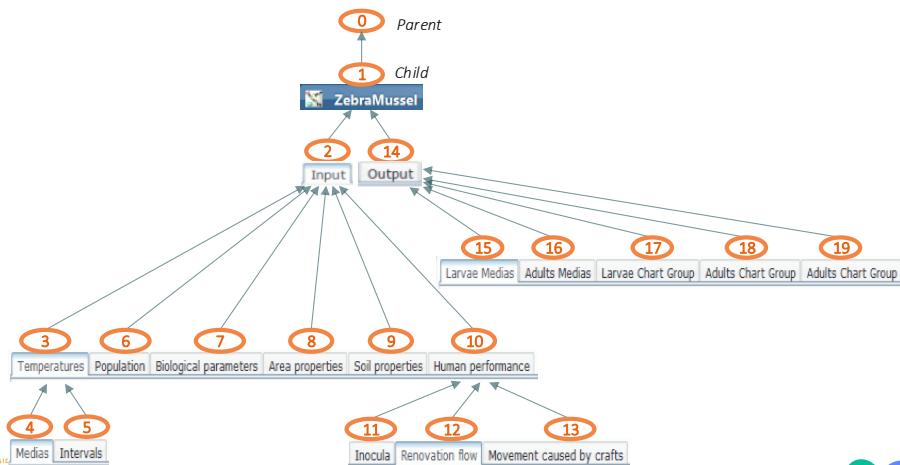
## Process

- Writing P-Lingua model (.pli).
- Setting MeCoSim app config file (.xls).
- Debugging (model, translation, app, simulator).
- Ready to enter scenarios (.ec2) and **experimentally validate** with experts.
- Once it is validated, **run virtual experiments**.

# MeCoSim app config I

Tab Id	Tab Name	Tab Parent Id
1	Zebra Mussel	0
2	Input	1
3	Temperatures	2
4	Averages	3
5	Intervals	3
6	Population	2
7	Biological parameters	2
8	Area properties	2
9	Soil properties	2
10	Human performance	2
11	Inocula	10
12	Movement caused by crafts	10
13	Movement larvae renovation	10
14	Output	1
15	Larvae Medias	14
16	Adults Medias	14
17	Larvae Chart	14
18	Adults Chart	14
19	AdultsPerAgeChart	14

# MeCoSim app config II



# MeCoSim app config III

Column Id	Column Name	Default Value	Editable	Tooltip
1	Zone		TRUE	Zone
2	Width (m)		TRUE	Width (m)
3	Length (m)		TRUE	Length (m)
4	Depth (m)		TRUE	Depth (m)
5	Capacity (m3)		TRUE	Capacity (m3)
6	Soil Type 1		TRUE	Soil Type 1
7	Soil Type 2		TRUE	Soil Type 2
8	Soil Type 3		TRUE	Soil Type 3
9	Soil Type 4		TRUE	Soil Type 4

# MeCoSim app config IV

**ZebraMussel**

Scenario Edit Model Simulation View Plugins Help

Input Output Debug console

Temperatures Population Biological parameters Area properties Soil properties Human performance

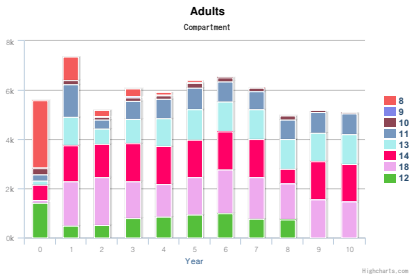
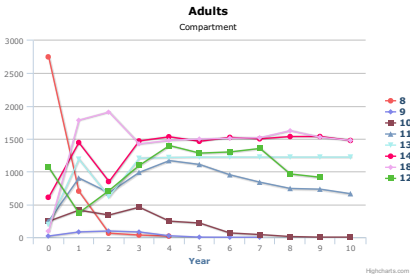
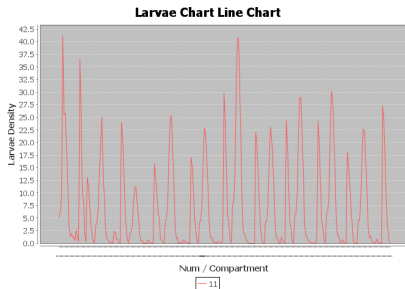
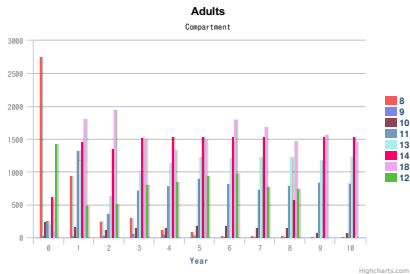
Zone	Width (m)	Length (m)	Depth (m)	Capacity (m3)	Soil Type 1	Soil Type 2	Soil Type 3	Soil Type 4
C71	266	2373	3	1741291	19	24	50	7
C51	468	2006	3	2913542	10	9	5	76
C111	545	3175	3,5	5966810	11	7	10	72
C81	346	7144	5	12785297	24	11	10	55
C41	314	8789	6,5	18081094	20	5	15	60
C21	408	3743	8	12638433	30	15	10	45
C11	412	3046	10	12549520	25	11	18	46
C72	266	2373	5	3308453	9	55	26	10
C52	468	2006	6	5535730	8	15	11	66
C112	545	3175	6,5	11336940	12	19	5	64
C82	346	7144	10	24292063	15	23	4	58
C42	314	8789	12,5	34354080	17	20	8	55
C22	408	3743	16	24013023	15	12	14	59
C12	412	3046	20	25099040	20	16	12	52
C61	566	2491	2	2819812	5	50	20	25
C62	566	2491	0	0	0	0	0	100
C31	272	2313	5	3254152	35	13	7	45
C32	272	2313	10	6182888	22	28	14	36

**P SYSTEM USER**  
**Scenario Data:** C:\Users\propietario\Dropbox\TESIS\Mussel Study\MeCoSim\muscleOutput.ec2  
**Model:** C:\Users\propietario\Dropbox\TESIS\Mussel Study\MeCoSim\Actualizado\zebra\_mussel.pli  
Simulated cycles: 1  
Simulations by cycle: 1  
Steps by cycle: 102  
Selected simulator: dndp4

0%

(c) 2011 Research Group on Natural Computing. <http://www.gcn.us.es>

# MeCoSim app config V



- 1 Introduction
- 2 Core
- 3 Applications
- 4 Conclusions**

# Conclusions I

## Conclusions

- Human beings **need to solve problems** makes us move forward.
- Unconventional Computing (particularly Membrane Computing) is **promising to overcome limitations** of electronic technology.
- **Software tools** are needed so far to simulate the abstract machines, and can aid theorists **to deepen in the study** of these models of computation.
- They can also be a **practical tool for end users to solve problems** (theoretical, real-life).



# Conclusions II

## Main contributions I

- Design of a **methodology** for solutions based on P systems.
- Development of **MeCoSim**, supporting the methodology and helping P systems designers and end users.
- Design and implementation of a plugins **architecture for the extensibility** of MeCoSim.
- Development and integration of a number of **MeCoSim plugins**.
- Design of a **system of repositories** for the community to share plugins, apps, models and scenarios.

# Conclusions III

## Main contributions II

- Design of a **language for simple kernel P systems** in **P-Lingua** framework.
- Development of **parser and simulator** inside **P-Lingua** framework.
- **Application** of the methodology and the developed tools **to solve problems** (theoretical and real-life).
- Development of **MeCoSim web site**.

# Conclusions IV

MeCoSim web site

## MeCoSim Membrane Computing Simulator



### MeCoSim installation



Java 1.7 required; Included in Path

For **Windows and Unix**, a MeCoSim shortcut will be created in your desktop.  
For **Mac OS X** users, run from MeCoSim dir: "CreateShortcut\_Mac.command".

Get Started

## Overview

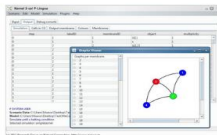
**MeCoSim** (Membrane Computing Simulator) is a software that offers the users analyze and verify different types of models based on **P systems**. Some of the

- **Simulation** of models of P systems under different initial conditions. It enables the load of P-Lingua based models, parsing, edition, **debugging**, and different simulation types.
- **Visualization** capabilities and graph
- **Highly customizable** for each model of a P system
- **Repositories** systems including **plugins**, c
- **Export** option for rel abstracting P system
- **Plugins** architecture external non-Java pr
- **Auto-update** capabilities

## Getting started

The best resources to start using MeCoSim and functionalities:

- User guide
  - Quick start
    - Installation and first use
    - Custom apps
    - Models
    - Scenarios
  - Working with models and simulations
    - Debug
    - Simulation
    - Repositories management
    - Plugins



## Case studies

MeCoSim development started in 2010, and a different areas of interest and application domain studies, grouped by areas. The examples concern MeCoSim. The number of scenarios and the references to related publications, charts and

- Case studies
  - Cell like P systems
  - Tissue like P systems
  - Spiking neural P systems
  - Population Dynamics P systems

study and y



# Research groups using MeCoSim I

- Department of Mathematics, Faculty of Life Sciences and Engineering, University of Lleida (Spain)
- Bearded Vulture Study and Protection Group, University of Lleida (Spain)
- Division of Conservation Biology, Institute of Ecology and Evolution, University of Berne Bern (Switzerland)
- Dirección de Medio Ambiente y Desarrollo Sostenible (Endesa)
- Department of Zoology, School of Natural Sciences, Trinity College Dublin (Ireland)
- Department of Animal Production (Division of Wildlife), Faculty of Life Sciences and Engineering, University of Lleida (Spain)



- Department of Computer Science, University of Sheffield (U.K.)
- Faculty of Mathematics and Computer Science , University of Bucharest (Romania)
- Faculty of Mathematics and Computer Science , University of Pitesti (Romania)
- Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan (China)
- School of Electrical Engineering, Southwest Jiaotong University, Chengdu (China)
- Center of Software Technology and Management Faculty of Information Science and Technology, University Kebangsaan Malaysia (Bangi, Malaysia)

- Methodology extensions: uncertainty, data analysis, *Data Science*.
- Simulation algorithms qualitative improvement: symbolic computation, RETE-like algorithms.
- Extension of the models of computation (e.g. *Fuzzy reasoning SNP systems*) covered (inside-outside *Membrane Computing*, within Artificial Intelligence).
- Optimization algorithms inclusion in **MeCoSim** (*Evolutionary Membrane Algorithms*, etc.).
- Visual definition of P systems: MDE (*Model Driven Engineering*), **Cell Designer** or internal development as in **MetaPLab**.
- Visual controls and buttons to enter scenario info (NetLogo).

- Output formats diversification: (**gnuplot**, **json**, etc.). In progress.
- Communication protocols standarization for external simulators.
- Non-connected capabilities (requesting simulations and receiving deferred results).
- General mechanism for functions definition for P-Lingua and parameters languages (JSR 223, or SAGE vision).
- Repositories visibility, automatic submission, community increase (Eclipse plugins, NetLogo models library, **R language** community).
- **MeCoSim** documentation improvement.

# Publications

ISI-JCR indexed journals

- 1 M.A. Colomer, A. Margalida, L. Valencia-Cabrera, A. Palau. Application of a computational model for complex fluvial ecosystems: the population dynamics of zebra mussel *Dreissena polymorpha* as a case study. *Ecological Complexity*, **20** (2014), 116-126. Impact factor: **2.000**
- 2 L. Valencia-Cabrera, M. García-Quismondo, M.J. Pérez-Jiménez, Y. Su, H. Yu, L. Pan. Modeling Logic Gene Networks by Means of Probabilistic Dynamic P Systems. *International Journal of Unconventional Computing*, **9**, 5-6 (2013), 445-464. Impact factor: **0.684**
- 3 M. García-Quismondo, L.F. Macías-Ramos, M.A. Martínez-del-Amor, I. Pérez-Hurtado, L. Valencia-Cabrera. Open Problems on Simulation of Membrane Computing Models. *International Journal of Foundations of Computer Science*, **24**, 5 (2013), 619-622. Impact factor: **0.554**
- 4 M. Gheorghe, F. Ipate, R. Lefticaru, M.J. Pérez-Jiménez, A. Turcanu, L. Valencia-Cabrera, M. García-Quismondo, L. Mierla. 3-COL problem modelling using simple Kernel P systems. *International Journal of Computer Mathematics*, **90**, 4 (2013), 816-830. Impact factor: **0.542**
- 5 D. Orellana-Martín, C. Graciani, L.F. Macías-Ramos, M.A. Martínez-del-Amor, A. Riscos-Núñez, Á. Romero-Jiménez, L. Valencia-Cabrera. Sevilla Carpets Revisited: Enriching the Membrane Computing Toolbox. *Fundamenta Informaticae*, **134** (2014), 153-166. Impact factor: **0.479**
- 6 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. Impact factor: **0.453**



# Publications

## Non-indexed journals

- 1 M.A. Colomer, S. Lavín, I. Marco, A. Margalida, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy, E. Serrano, L. Valencia-Cabrera. Modeling population growth of Pyrenean Chamois (*Rupicapra p. pyrenaica*) by using P systems. *Lecture Notes in Computer Science*, **6501** (2011), 144-159.
- 2 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.
- 3 M.A. Martínez-del-Amor, I. Pérez-Hurtado, M. García-Quismondo, L.F. Macías-Ramos, L. Valencia-Cabrera, Á. 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.
- 4 L.F. Macías-Ramos, M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius. The efficiency of tissue P systems with cell separation relies on the environment. *Lecture Notes in Computer Science*, **7762** (2013), 243-256.
- 5 F. Ipate, R. Lefticar, L. Mierla, L. Valencia-Cabrera, H. Hang, G. Zhang, C. Dragomir, M.J. Pérez-Jiménez, M. Gheorghe. Kernel P systems: Applications and Implementations. *Advances in Intelligent Systems and Computing*, **212** (2013), 1081-1089.
- 6 M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius, L. Valencia-Cabrera. The relevance of the environment on the efficiency of tissue P systems. *Lecture Notes in Computer Science*, **8340** (2014), 308-321.
- 7 L.F. Macías-Ramos, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez, A. Riscos-Núñez, L. Valencia-Cabrera. The Role of the Direction in Tissue P Systems with Cell Separation. *Journal of Automata, Languages and Combinatorics*, **19**, 1-4 (2014), 185-199.

# Publications

## Book chapters

- 1 M. A. Colomer, M. García-Quismondo, L. F. Macías, M. A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez, L. Valencia-Cabrera. Membrane System-Based Models for Specifying Dynamical Population Systems. In P. Frisco, M. Gheorghe, M. J. Pérez-Jiménez (eds.) *Applications of Membrane Computing in Systems and Synthetic Biology*. Springer, Series: Emergence, Complexity and Computation, Vol. 7, 2014, Chapter 4, pp. 97-132.



- ① 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.) *Proceedings 2010 IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2010)*, IEEE Press, Volume 1, September 23-26, 2010, Changsha, China, pp. 637-643.
- ② M. Angels Colomer, C. Fondevilla, L. Valencia-Cabrera (2011). A New P System to Model the Subalpine and Alpine Plant Communities. *Proceedings of the Ninth Brainstorming Week on Membrane Computing*, Seville, Spain, January 31- February 4, 2011, Report RGNC 01/2010, Fénix Editora, 2011, pp. 91-112.
- ③ R. Lefticaru, F. Ipate, L. Valencia-Cabrera, 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. *Proceedings of the Tenth Brainstorming Week on Membrane Computing*, Volume I, Seville, Spain, January 30- February 3, 2012, Report RGNC 01/2012, Fénix Editora, 2012, pp. 291-318.
- ④ M. Gheorghe, F. Ipate, C. Dragomir, L. Mierla, L. Valencia-Cabrera, M. García-Quismondo, M.J. Pérez-Jiménez. Kernel P systems - Version I. *Proceedings of the Eleventh Brainstorming Week on Membrane Computing*, Seville, Spain, February 4-8, 2013, Report RGNC 01/2013, Fénix Editora, 2013, pp. 97-124.
- ⑤ L. Valencia-Cabrera, M. García-Quismondo, M.J. Pérez-Jiménez, Y. Su, H. Yu, L. Pan. Analising gene networks with PDP systems. Arabidopsis thaliana, a case study. *Proceedings of the Eleventh Brainstorming Week on Membrane Computing*, Seville, Spain, February 4-8, 2013, Report RGNC 01/2013, Fénix Editora, 2013, pp. 257-272.
- ⑥ M.A. Martínez-del-Amor, L.F. Macías-Ramos, L. Valencia-Cabrera, A. Riscos-Núñez, M.J. Pérez-Jiménez . Accelerated Simulation of P Systems on the GPU: A Survey. In L. Pan, Gh. Paun, M. J. Pérez-Jiménez, T. Song (eds.) *Bio-inspired Computing: Theories and Applications*. Series: Communications in Computer and Information Science, Volume 472, 2014, pp. 308-312.

- **Cell-based Membrane Computing systems and their applications in Biology.**  
*National Natural Science Foundation of China. Grant No. 61320106005*
- **De la Computación Celular a la Computación de alto rendimiento. Aplicación a la dinámica de poblaciones.**  
*Ministerio de Economía y Competitividad. TIN2012-37434*
- **Computación Celular: Aplicaciones a la Biología de Sistemas y Sintética.**  
*Ministerio de Ciencia e Innovación. TIN2009-13192*
- **Modelización y simulación computacional en Biología de Sistemas.**  
*Junta de Andalucía. Proyecto de Excelencia con Investigador de Reconocida Valía. P08-TIC-04200.*
- **Diseño y aplicación de un modelo PDP (Population Dynamics P Systems) para la estimación de la disponibilidad y distribución espacial de los recursos tróficos para aves carroñeras en Navarra.**  
*Gestión Ambiental, Viveros y Repoblaciones de Navarra, S.A. (Proyecto NECROPIR - EFA 130/09. Unión Europea. Cooperación territorial España-Francia-Andorra).*
- **Desarrollo de una aplicación interactiva para simular la gestión de un ecosistema formado por un río y un embalse gestionado por el hombre.**  
*Endesa S.A.*
- **Desarrollo, calibración y validación de un modelo de simulación del comportamiento de las poblaciones del mejillón cebra en embalses. Aplicación al embalse de Ribarroja.**  
*Endesa S.A.*

My sincere thanks

vă mulțumesc foarte mult pentru atenție

Thank you very much for your attention!

¡Muchas gracias por su atención!

