An Introduction to Membrane Computing

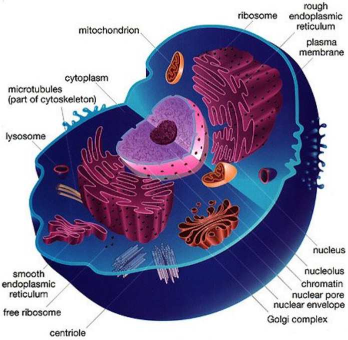
A. Riscos-Núñez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Universidad de Sevilla

3rd Int. School on Biomolecular and Biocellular Computing
June 28-30, Valencia

1. Introduction
2. (Cell-like) P Systems with Active Membranes
3. Tissue-like P Systems with Cell Division / Separation
4. Neural-like Approach

Can cells compute?



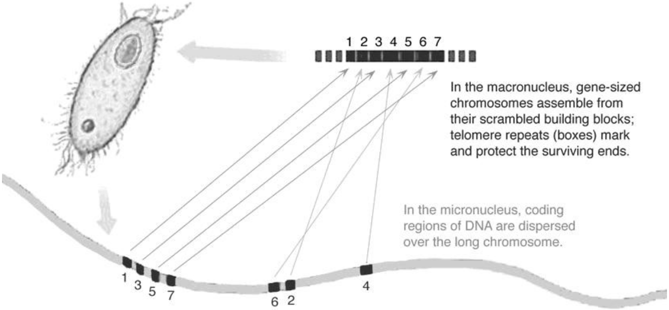
Computation is out there

Arithmetics

- They can count (threshold): *quorum sensing*
- They can distribute / divide: *mitosis*

Memory pointers

- Genes self-assembly in Cilliates



Membrane Computing

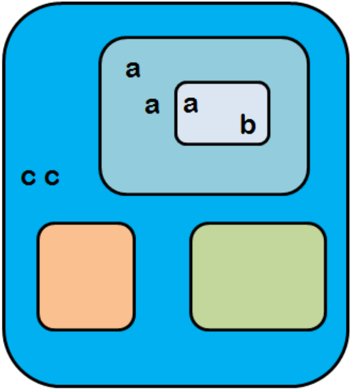


Figure: A P system

- Multisets of objects
  - distinguished alphabets
- Membranes (regions)
  - a.k.a. cells, neurons
  - input / output regions
- Rules
  - Objects
  - Membranes
- Environment

Membrane Computing

It has developed quickly into a **vigorous** scientific discipline.

- ⋆ International Conference on Membrane Computing (18th edition).
- ⋆ Brainstorming Week on Membrane Computing (15th edition).
- ⋆ Asian Conference on Membrane Computing (6th edition).

In 2003, Thomson Institute for Scientific Information (**ISI**) declared this area as a **Fast Emerging Research Front in Computer Science**

In 2016, **International Membrane Computing Society** was founded

It has developed quickly into a **vigorous** scientific discipline.

- ⋆ International Conference on Membrane Computing (18th edition).
- ⋆ Brainstorming Week on Membrane Computing (15th edition).
- ⋆ Asian Conference on Membrane Computing (6th edition).

In 2003, Thomson Institute for Scientific Information (**ISI**) declared this area as a **Fast Emerging Research Front in Computer Science**

In 2016, **International Membrane Computing Society** was founded

Objects

- strings, arrays, spikes, ...

Membranes

- tree-like / tissue-like structure
- labels, charges, proteins, ...

Semantics ingredients

- **Configuration** (Initial configuration, halting configuration)
- **Transition step** (how rules are applied)
- **Computation** (sequence of configurations: halting computation)

Rules

- selecting which **types**
  (e.g. forbidding dissolution, using only communication, . . .)
- controlling **applicability**
  (e.g. priorities, permitting / forbidding conditions,
  alternatives to maximal parallelism, . . .)

Diversity of interpretations

- *Generative* devices: fixed initial configuration, we collect the outputs of all the non-deterministic computations.
- *Computing* devices: given an input (encoded somehow), compute the resulting output (multiset).
- *Decision* tools: special objects *yes* and *no*, s.t. their presence / absence in the output decides whether the given input was accepted by the P system or not.
- *Simulation* tools: no halting configuration, the output is the computation.

Diversity of interpretations

- *Generative* devices: fixed initial configuration, we collect the outputs of all the non-deterministic computations.
- *Computing* devices: given an input (encoded somehow), compute the resulting output (multiset).
- *Decision* tools: special objects *yes* and *no*, s.t. their presence / absence in the output decides whether the given input was accepted by the P system or not.
- *Simulation* tools: no halting configuration, the output is the computation.

Diversity of interpretations

- *Generative* devices: fixed initial configuration, we collect the outputs of all the non-deterministic computations.
- *Computing* devices: given an input (encoded somehow), compute the resulting output (multiset).
- *Decision* tools: special objects *yes* and *no*, s.t. their presence / absence in the output decides whether the given input was accepted by the P system or not.
- *Simulation* tools: no halting configuration, the output is the computation.

Diversity of interpretations

- *Generative* devices: fixed initial configuration, we collect the outputs of all the non-deterministic computations.
- *Computing* devices: given an input (encoded somehow), compute the resulting output (multiset).
- *Decision* tools: special objects *yes* and *no*, s.t. their presence / absence in the output decides whether the given input was accepted by the P system or not.
- *Simulation* tools: no halting configuration, the output is the computation.

Main research directions

- Theoretical Foundations
  - **Universality** results
    Generative / accepting power equivalent to . . .
  - What if . . . ?
  - Formalization
- Computational Complexity
  - **Efficient** solutions to **hard** problems
  - **P conjecture**
- Practical Approach
  - **Simulators**
  - **Modelling**
  - Generative music, Robot control, Model checking, . . .

Initial models: Cell-like approach

Based on a hierarchical arrangement of membranes delimiting **compartments** where multisets of chemicals **evolve** according to given evolution rules.

- The rules are either modeling chemical reactions (in the form of multiset rewriting rules), or they are inspired by other biological processes (passing objects through membranes, mitosis, etc.) and have the form of communication rules, division rules, etc.

Initial models: Cell-like approach

- $\Pi = (\Gamma, \Sigma, H, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$.
- *Basic transition* P systems:
  - $[\ u\ ]_h \rightarrow [\ v\ ]_h$ (evolution rules).
  - $[\ u\ ]_h \rightarrow v\ [\ \ ]_h$ and $u\ [\ \ ]_h \rightarrow [\ v\ ]_h$ (communication rules).
  - $[\ u\ ]_h \rightarrow v$ (dissolution rules).
- $\mathcal{T}$: class of recognizer basic transition P systems.

Initial models: Cell-like approach

- $\Pi = (\Gamma, \Sigma, H, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$.
- *Basic transition* P systems:
  - $[\ u\ ]_h \to [\ v\ ]_h$ (evolution rules).
  - $[\ u\ ]_h \to v\ [\ \ ]_h$ and $u\ [\ \ ]_h \to [\ v\ ]_h$ (communication rules).
  - $[\ u\ ]_h \to v$ (dissolution rules).
- $\mathcal{T}$: class of recognizer basic transition P systems.

P systems with active membranes

(*a*) $[\, a \to u \,]_h^{\alpha}$ (*object evolution* rules).

(*b*) $a\,[\ ]_h^{\alpha_1} \to [\, b \,]_h^{\alpha_2}$ (*send–in communication* rules).

(*c*) $[\, a \,]_h^{\alpha_1} \to [\ ]_h^{\alpha_2}\, b$ (*send–out communication* rules).

(*d*) $[\, a \,]_h^{\alpha} \to b$ (*dissolution* rules).

(*e*) $[\, a \,]_h^{\alpha_1} \to [\, b \,]_h^{\alpha_2}\, [\, c \,]_h^{\alpha_3}$ (*division* rules for *elementary membranes*).

(*f*) $[\, [\,]_{h_1}^{\alpha_1}\ [\,]_{h_2}^{\alpha_2}\,]_h^{\alpha} \to [\, [\,]_{h_1}^{\alpha_3}\,]_h^{\beta}\ [\, [\,]_{h_2}^{\alpha_4}\,]_h^{\gamma}$ (*division* rules for *non–elementary membranes*).

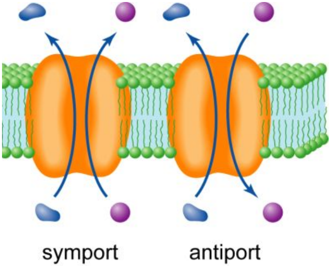
Tissue-like P systems

Inspired by

- intercellular communication
- cooperation between neurons

- Communication rules: *symport/antiport*
- Cells as nodes of a graph (and environment)

Tissue-like P systems



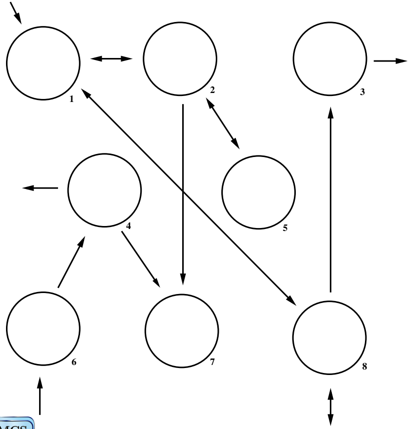
Based on the complex communication networks established among adjacent cells by making their protein channels cooperate, moving molecules directly from one cell to another.

Tissue-like P systems

Symport/antiport rules define a directed graph in an **implicit** way.

Rules

$(0, ba^2/\lambda, 1), (0, \lambda/b^4cd, 3),$
$(0, \lambda/ab^3, 4), (0, c/\lambda, 6),$
$(0, a/b^2, 8), (1, c^3/b^2, 2)$
$(1, ad/a, 8), (2, ab/\lambda, 7),$
$(2, b/b^2, 5), (3, \lambda, d^2, 8),$
$(4, \lambda/a, 6), (4, b^2c^2/\lambda, 7)$



Tissue-like P systems with Cell Division / Separation

Finite alphabets

- working alphabet (Γ),
- **input** alphabet ($\Sigma \subseteq \Gamma$),
- **environment** alphabet ($\mathcal{E} \subseteq \Gamma \setminus \Sigma$)

Rules

- symport/antiport: $(i, u/v, j)$
- cell division: $[a]_i \rightarrow [b]_i[c]_i$
- cell separation: $[\ a\ ]_i \rightarrow [\ \Gamma_1\ ]_i[\ \Gamma_2\ ]_i$ (for a fixed partition)

Length of rule $(i, u/v, j)$ = $|u| + |v|$

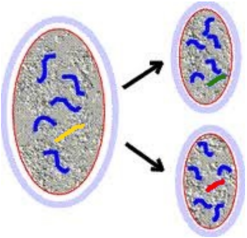
Tissue-like P systems with Cell Division / Separation

Initial configuration

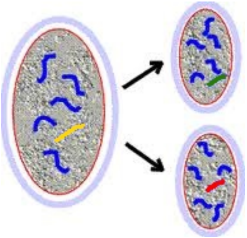
- multisets $\mathcal{M}_1, \ldots, \mathcal{M}_q$ over $\Gamma \setminus \Sigma$
- environment

- Non deterministic, maximally parallel.
- While dividing / separating, communication is blocked.
- Division $\Rightarrow$ duplicate and transfer to the new cells.
- Separation $\Rightarrow$ objects are distributed among the new cells.

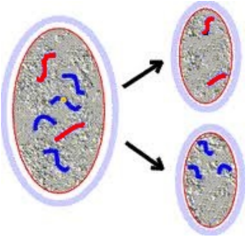
- *Cell Division*



- *Cell Division*



- *Cell Separation*



Cell Division: $[a]_i \rightarrow [b]_i[c]_i$

Contents duplicated

$$\boxed{a^5\ \ b\ \ c^2\ \ d^3}$$

Cell Division: $[a]_i \to [b]_i[c]_i$

Contents duplicated

$a^4\ b^2\ c^2\ d^3$

$a^4\ b\ c^3\ d^3$

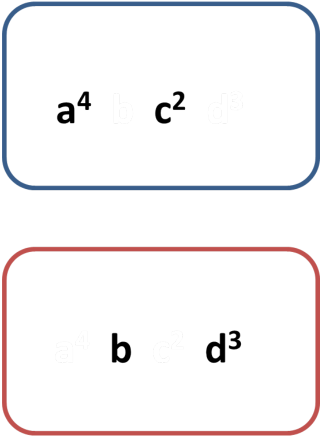
Cell Separation: $[a]_i \rightarrow [\Gamma_1]_i[\Gamma_2]_i$

Contents distributed

$a^5 \ \ b \ \ c^2 \ \ d^3$

Cell Separation: $[a]_i \rightarrow [\Gamma_1]_i[\Gamma_2]_i$

Contents distributed



- Optimal watermarking[1]
    - Digital watermarking has became one of the most effective tools for copyright protection of digital media (such as image, audio, video).
- Image segmentation[2]
    - Image segmentation is one of the most important problems in computer vision and video applications.
    - A fast multi-level thresholding method that uses the fuzzy entropy as the evaluation criterion.

[1] H. Peng, J. Wang, M.J. Pérez-Jiménez, A. Riscos. The framework of P systems applied to solve optimal watermarking problem. Signal Processing, 101 (2014), 256-265.

[2] H. Peng, J. Wang, M.J. Pérez-Jiménez, P. Shi. A novel image thresholding method based on membrane computing and fuzzy entropy. Journal of Intelligent and Fuzzy Systems, 24, 2 (2013), 229-237.

Neural-like approach

Biological motivation based on recent discoveries on neural coding

- Various studies show evidence of precise temporal correlations between pulses of different neurons and stimulus-dependent synchronisation of the activity in populations of neurons (see, e.g., Eckhorn et al. 1988[a] or Gerstner and Kistler 2002[b])

[a] R. Eckhorn, R Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, H.J. Reitboeck. Coherent oscillations: a mechanism of feature linking in the visual cortex? Biol Cybern. 60 (1988), 121-130.

[b] W. Gerstner, W. Kistler. Spiking neuron models. Single neurons, populations, plasticity. Cambridge University Press, Cambridge, MA, 2002.

Informally, an SN P system consists of a set of neurons placed in the nodes of a directed graph which send signals (called *spikes*) along the arcs of the graph (representing the *synapses* between neurons).

Neural-like approach

- The system evolves according to a set of *spiking rules* and *forgetting rules* (for sending or internally consuming spikes)
- Applicability w.r.t. all spikes present in a neuron (although maybe not all of them consumed)
- The produced spikes are sent (maybe with a delay of some steps) via all outgoing synapses
- In each time unit each neuron which can use a rule should use **one** such rule
- One of the neurons is considered to be the output neuron, and its spikes are also sent to the environment.

Neural-like approach

- The system evolves according to a set of *spiking rules* and *forgetting rules* (for sending or internally consuming spikes)
- Applicability w.r.t. all spikes present in a neuron (although maybe not all of them consumed)
- The produced spikes are sent (maybe with a delay of some steps) via all outgoing synapses
- In each time unit each neuron which can use a rule should use **one** such rule
- One of the neurons is considered to be the output neuron, and its spikes are also sent to the environment.

Neural-like approach

- The system evolves according to a set of *spiking rules* and *forgetting rules* (for sending or internally consuming spikes)
- Applicability w.r.t. all spikes present in a neuron (although maybe not all of them consumed)
- The produced spikes are sent (maybe with a delay of some steps) via all outgoing synapses
- In each time unit each neuron which can use a rule should use **one** such rule
- One of the neurons is considered to be the output neuron, and its spikes are also sent to the environment.

Neural-like approach

- The system evolves according to a set of *spiking rules* and *forgetting rules* (for sending or internally consuming spikes)
- Applicability w.r.t. all spikes present in a neuron (although maybe not all of them consumed)
- The produced spikes are sent (maybe with a delay of some steps) via all outgoing synapses
- In each time unit each neuron which can use a rule should use **one** such rule
- One of the neurons is considered to be the output neuron, and its spikes are also sent to the environment.

Neural-like approach

- The system evolves according to a set of *spiking rules* and *forgetting rules* (for sending or internally consuming spikes)
- Applicability w.r.t. all spikes present in a neuron (although maybe not all of them consumed)
- The produced spikes are sent (maybe with a delay of some steps) via all outgoing synapses
- In each time unit each neuron which can use a rule should use **one** such rule
- One of the neurons is considered to be the output neuron, and its spikes are also sent to the environment.

Spiking Neural P systems

$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$

- A singleton alphabet $O = \{a\}$: spike.
- A set of cells (neurons).
- For each neuron $\sigma_i = (n_i, R_i)$:
  - n_i: initial number of spikes.
  - R_i: set of developmental rules.
    - *Spiking rules*: $E/a^c \to a; d$, where E is a regular expression over a, $c \geq 1$, and $d \geq 0$;
    - *Forgetting rules*: $a^s \to \lambda$, for some $s \geq 1$, with the restriction that for each spiking rule $E/a^c \to a; d$ from R_i, we have $a^s \notin L(E)$;
- A structure syn (synapse graph).
  - Arcs: Specify spikes flow among neurons.
- Two distinguished neurons: input, output (connected to the environment).

Spiking Neural P systems

$\Pi = (O, \sigma_1, \dots, \sigma_m, syn, in, out)$

- A singleton alphabet $O = \{a\}$: spike.
- A set of cells (neurons).
- For each neuron $\sigma_i = (n_i, R_i)$:
  - n_i: initial number of spikes.
  - R_i: set of developmental rules.
    - Spiking rules: $E/a^c \to a; d$, where E is a regular expression over a, $c \geq 1$, and $d \geq 0$;
    - Forgetting rules: $a^s \to \lambda$, for some $s \geq 1$, with the restriction that for each spiking rule $E/a^c \to a; d$ from R_i, we have $a^s \notin L(E)$;
- A structure syn (synapse graph).
  - Arcs: Specify spikes flow among neurons.
- Two distinguished neurons: input, output (connected to the environment).

Spiking Neural P systems

$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$

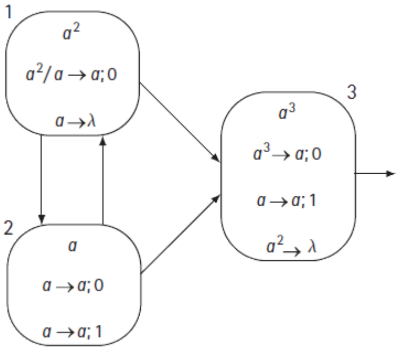
- A singleton alphabet $O = \{a\}$: spike.
- A set of cells (neurons).
- For each neuron $\sigma_i = (n_i, R_i)$:
  - n_i: initial number of spikes.
  - R_i: set of developmental rules.
    - *Spiking rules*: $E/a^c \to a; d$, where E is a regular expression over a, $c \geq 1$, and $d \geq 0$;
    - *Forgetting rules*: $a^s \to \lambda$, for some $s \geq 1$, with the restriction that for each spiking rule $E/a^c \to a; d$ from R_i, we have $a^s \notin L(E)$;
- A structure syn (synapse graph).
  - Arcs: Specify spikes flow among neurons.
- Two distinguished neurons: input, output (connected to the environment).

Spiking Neural P systems

$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$

- A singleton alphabet $O = \{a\}$: spike.
- A set of cells (neurons).
- For each neuron $\sigma_i = (n_i, R_i)$:
  - n_i: initial number of spikes.
  - R_i: set of developmental rules.
    - *Spiking rules*: $E/a^c \rightarrow a; d$, where E is a regular expression over a, $c \geq 1$, and $d \geq 0$;
    - *Forgetting rules*: $a^s \rightarrow \lambda$, for some $s \geq 1$, with the restriction that for each spiking rule $E/a^c \rightarrow a; d$ from R_i, we have $a^s \notin L(E)$;
- A structure *syn* (synapse graph).
  - Arcs: Specify spikes flow among neurons.
- Two distinguished neurons: input, output (connected to the environment).

An example



An SN P system generating all natural numbers greater than 1.

Applications of SN P systems

- Weighted Fuzzy Spiking Neural P systems[3]
  - Are able to express fuzzy and uncertain knowledge and to process weighted fuzzy reasoning.
  - A Weighted Fuzzy reasoning algorithm based on WFSN P systems.
  - Share some common features with Petri nets but this is asynchronous.
- Fuzzy Reasoning Spiking Neural P systems[4]
  - Model fuzzy fuzzy production rules in a fuzzy diagnosis knowledge base.
  - A parallel fuzzy reasoning algorithm based on FRSN P systems.
  - Fault diagnosis of electric locomotive systems (with real number)[5].
  - Fault diagnosis in power transmission networks for single and multiple fault situations with/without incomplete and uncertain SCADA data (with trapezoidal number)[6] .

---

[3] J. Wang, P. Shi, H. Peng, M.J. Pérez-Jiménez, T. Wang. Weighted Fuzzy Spiking Neural P Systems. IEEE Transactions on Fuzzy Systems, 21, 2 (2013), 209-220.

[4] H. Peng, J. Wang, M.J. Pérez-Jiménez, H. Wang, J. Shao, T. Wang. Fuzzy reasoning spiking neural P system for fault diagnosis. Information Sciences, 235 (2013), 106-116.

[5] T. Wang, G. Zhang, M.J. Pérez-Jiménez. Fault diagnosis models for electric locomotive systems based on Fuzzy Reasoning Spiking Neural P Systems. Proceedings of the 15th International Conference on Membrane Computing (M. Gheorghe et al. eds.). Prague, Czech Republique, August, 20-22, 2014, pp. 361-374.

[6] T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang, M.J. Pérez-Jiménez. Fault Diagnosis of Electric Power Systems Based on Fuzzy Reasoning Spiking Neural P Systems. IEEE Transactions on Power Systems, online version, 2014.