How Membrane Computing Influences Theoretical Computer Science?

Erzsébet Csuhaj-Varjú

Department of Algorithms and Applications,
Faculty of Informatics,
Eötvös Loránd University, Budapest, Hungary

csuhaj@inf.elte.hu

Contents

- Theoretical Computer Science, the research area
- Unconventional Computing, Natural Computing – new approaches to computation
- Membrane Computing, motivations, main variants, research topics and areas, results
- Impact of Membrane Computing on Theoretical Computer Science, the whole
- P automata, an example for a possible impact
- Conclusions, open problems, suggestions

Theoretical Computer Science (TCS)

Theoretical Computer Science is a **research area** that belongs to both *computer science* (general computer science) and *mathematics.*

It **focuses on** (more) **mathematical topics** of computing and includes the **theory of computation**.

Subfields of theoretical computer science, thus treat problems with *mathematical rigour*.

Theoretical Computer Science

Research areas that belong to theoretical computer science are, among others:

Algorithms,

Data structures,

Foundations of Computing,

Computational complexity,

Parallel and distributed computation,

Information theory,

Cryptography,

Program semantics and verification,

Machine learning,

Computational biology,

Computational economics, etc.

(Classical) Computing

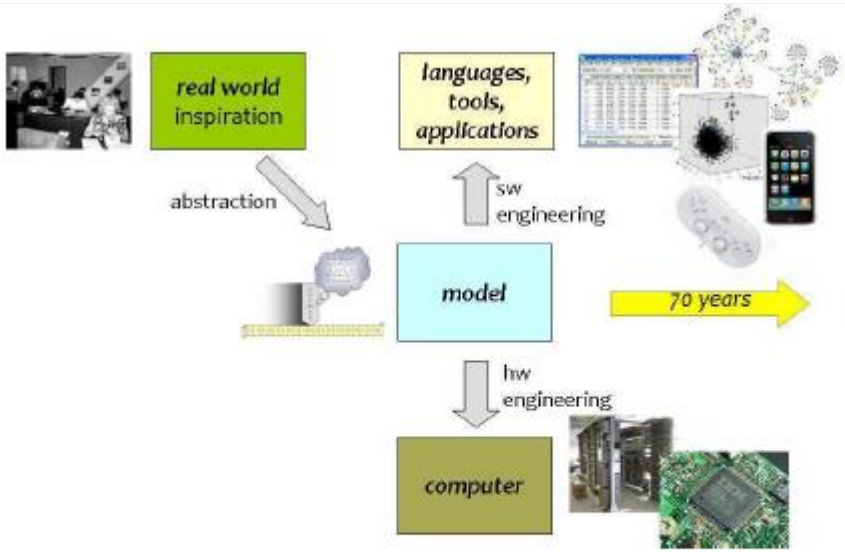


Figure 1. Classical computation: the real world inspiration of human computers led to an abstract model. This was realised in hardware and exploited in software, and developed for 70 years, into a form unrecognisable to its early developers.

[S. Stepney, 2012]

Unconventional Computing (Aims)

- To create **non-standard** computational models that „go beyond" **Turing machines** and the **von Neumann's architecture**

- To understand better what

- ***computation,***
- ***information processing and information flow,***
- ***dynamical behaviour,***
- ***chaotic behaviour,***
- ***development,***
- ***self-reproduction, etc.*** *mean.*

Classical versus Unconventional Computing

Classical computation got things backwards:

theory before hardware and applications

Unconventional computing takes different routes:

The real word inspiration leads to novel hardware (in some cases wetware), rather than directly to a model

Natural Computing

'' A field of research that investigates models and computational techniques inspired by nature and, dually, attempts to understand the world around us in terms of information processing.''

[Kari, Rozenberg, 2008, Communication of the ACM]

Natural phenomena as motivation

self-reproduction,

functioning of the brain,

characteristics of life,

group behavior,

cell membranes,

tissue organization

etc.

Some Characteristics of Natural Processes

Natural processes (bio-processes) can be considered as **computational processes**.

- Usually, the **object of the computation** and the **„machine”** which executes the computation **cannot be separated**.

- The „machine” can use for computation **only existing objects**.

Some Characteristics of Natural Processes

- Often, **natural processes** are **not composed** from a **set of elements bounded by a constant**.
- There may be natural processes **not bounded in time** (**infinite run is possible**).
- Most of **natural systems** are **complex systems**.

Nature-inspired Computing

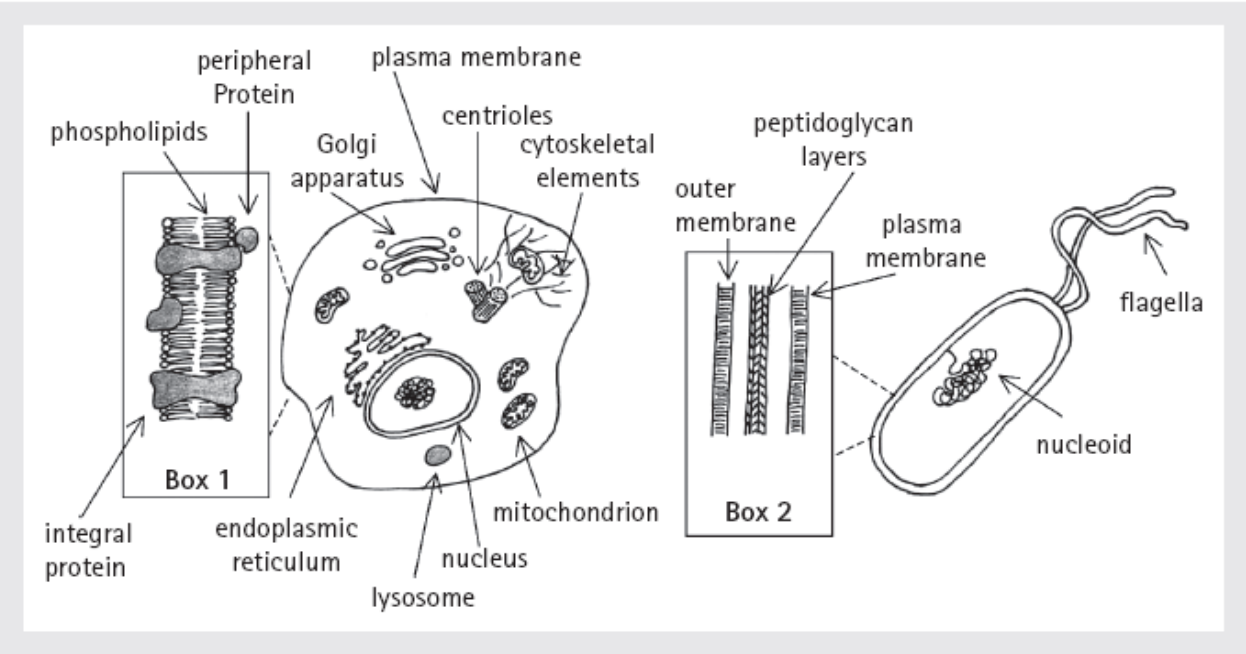
- Cellular automata
- Neural networks
- Evolutionary computation
- Swarm intelligence
- Artificial life
- **Membrane computing (MC)**

Membrane Systems (P systems)

Computational models abstracted from the **architecture** and the **functioning** of the living **cells** and **tissues.**

(Gheorghe Paun, 1998)

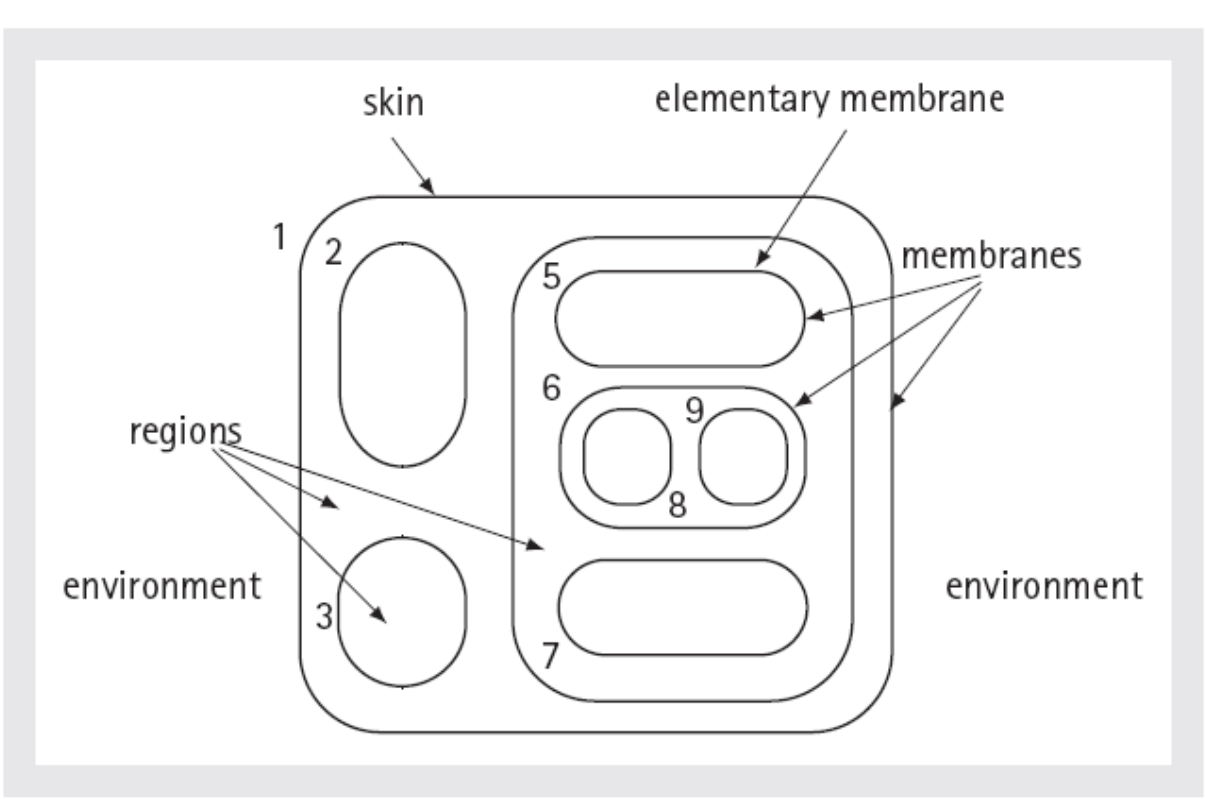
P Systems, Motivation – Cell



(The Oxford Handbook of Membrane Computing,
Gh. Paun, G. Rozenberg, A. Salomaa, eds., Oxford University Press, 2010)

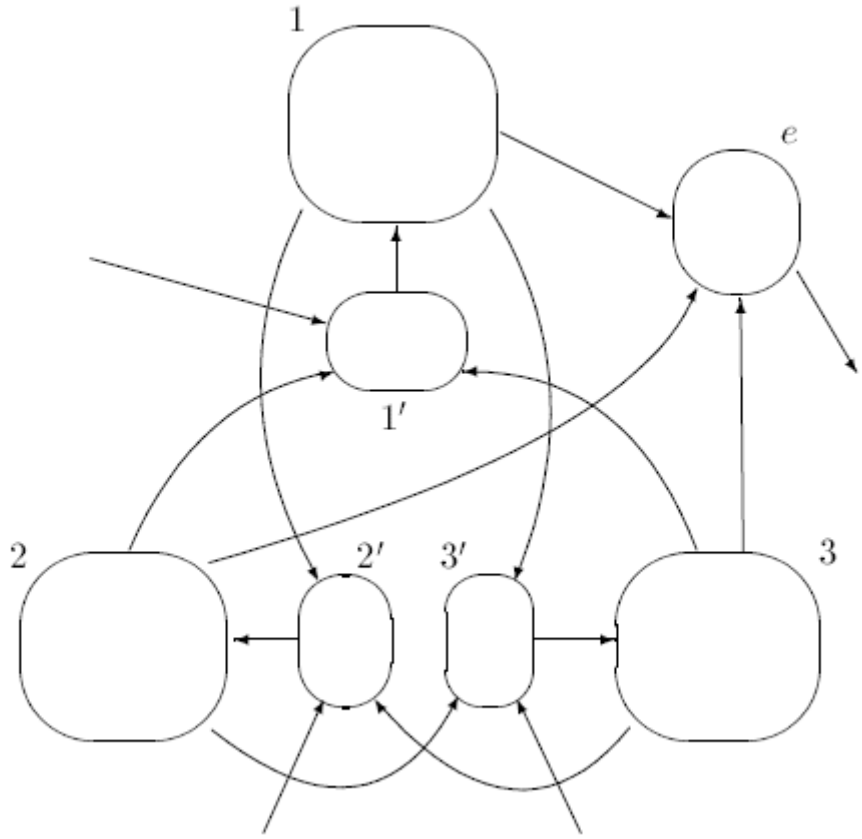
Membrane Structure

A hierarchical arrangement of regions where multisets of objects evolve according to given evolutionary rules



(The Oxford Handbook of Membrane Computing,
Gh. Paun, G. Rozenberg, A. Salomaa, eds., Oxford University Press, 2010)

Tissue-like P Systems



The underlying structure is an arbitrary virtual graph, different variants of communication are considered

Membrane Systems, Multiset Rewriting Rules

$a^2bc^3 \rightarrow ba^2c(da,out)(ca,in)$

The **rules**

change the objects

move the objects between neighbouring regions

The rules are applied

in **parallel**

in a **synchronized** manner

P System – the Basic Variant

$$\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_o),$$

where:

(i) O is an alphabet – its elements are called *objects*;

(ii) μ is a membrane structure consisting of m membranes, with the membranes (and hence the regions) injectively labeled with $1, 2, \ldots, m$; m is called the *degree* of Π;

(iii) $w_i, 1 \leq i \leq m$, are strings which represent multisets over V associated with the regions $1, 2, \ldots, m$ of μ;

(iv) $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* over O; R_i is associated with the region i of μ; an *evolution rule* is of the form $u \rightarrow v$, where u is a string over O and v is a string over O_{tar}, where $O_{tar} = O \times TAR$, for $TAR = \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$;

(v) $i_o \in \{1, 2, \ldots, m\}$ is the label of an elementary membrane (the *output membrane*).

Computation by a P System

- The system starts in an **initial configuration**, and **evolves** according to its rules,
- by **changing**, ***creating***, ***deleting***, and **moving** the objects between the regions (nodes) in parallel.
- Some of the **evolutions/computations** are defined to be **successful** (no rule is applicable in any of the regions (nodes), a final configuration is reached, etc.), and these yield
- a **result** (a *number* or a *vector of multiplicities* of objects in the regions (nodes) or in the environment, etc.)

P Systems

- **Main components**
  - Objects
  - Rules
  - Underlying graph (architecture)
- **Other main characteristics**
  - Type of rule application
  - Mode of use (generating, accepting)
  - Type of result

Variants of P Systems

- **Objects**: symbols, strings, spikes, arrays, trees, …
- **Data structures**: multisets, sets (languages)
- **Location of objects**: in the regions, in the nodes, on the membranes, on the edges, combined cases
- **Form of rules**: multiset rewriting rules, (purely) communication rules, rules with membrane creation, division, dissolution, spike processing, etc.

Variants of P Systems - continued

- **Control of application of rules**: catalysts, priority, promoters, inhibitors, channels, etc.
- **Membrane configurations:** cell-like (tree), tissue-like (arbitrary graph), static or dynamic communication channels (population P systems)
- **Type of the membrane structure**: static, dynamic, precomputed
- **Timing:** synchronized, asynchronous, time-free, etc.

Variants of P systems - continued

- **Application of rules**: maximal parallelism, minimal parallelism, bounded parallelism, sequential, etc.
- **Successful computations**: global halting, local halting, etc.
- **Modes of using the system**: generating, accepting
- **Types of output**: set of numbers, set of vectors of numbers, languages, *yes/no* answer

Research Directions/Issues

- computing power, computational efficiency,
- descriptional complexity, normal forms, hierarchies,
- algorithms,
- modelling,
- implementations,
- simulations,
- semantics,
- model checking, verifications,
- relations to dynamical systems,
- etc…

Types of Results (MC & TCS results)

- universality, computational power, hypercomputation
- collapsing hierarchies, infinite hierarchies,
- normal forms,
- polynomial solutions to NP-complete problems and even to PSPACE-complete problems (with time/space tradeoff),
- classifications, comparisons with Chomsky and Lindenmayer hierarchies,
- comparisons with classic complexity classes, new complexity classes
- new algorithms for distributed and parallel systems,
- membrane algorithms,
- tools for modelling, model checking, verification, etc.

Types of Applications

- biology/biomedicine,
- population dynamics, ecosystems,
- economics,
- optimization,
- computer graphics,
- linguistics, natural language processing
- computer science,
- cryptography

Links to Other Models in (T)CS

- Petri nets,
- process algebra,
- X-machines,
- lambda calculus,
- ambient calculus, brane calculi

P Systems versus Classical Models in TCS

Standard features/characteristics (also in TCS)

distribution, communication, modularity,
dynamic change of structure, etc.

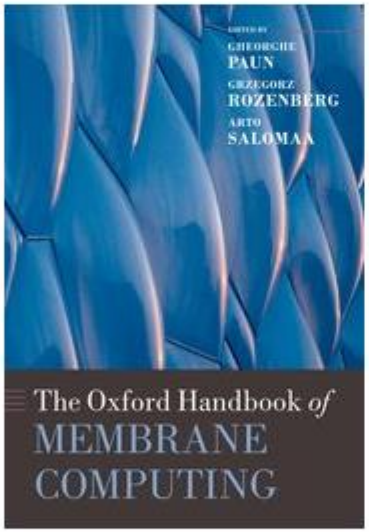
Unconventional features:

unbounded, massive parallelism, **properties which mimic properties of natural systems**.

Multiset rewriting systems

Membrane Computing (MC): impacts on Theoretical Computer Science TCS)

- MC contributes **to better understand the nature** of (classic) **computational models, algorithms, computing**.(?)
- MC provides **better/more efficient tools** to solve **problems of TCS that have been solved**.(?)
- MC provides **tools for unsolved problems** of TCS (?)
- MC **provides new, more efficient, more complex models**, **equivalent to classical models** of TCS. (?)
- MC **provides models „going beyond Turing”**.



The Oxford Handbook of Membrane Computing

Edited by **Gheorghe Paun**, **Grzegorz Rozenberg**, and **Arto Salomaa**

Oxford Handbooks

- Comprehensive presentation of membrane computing
- Professionally written, by the leading authors in this area
- Covers both theory and application
- Comprehensive bibliography
- Full background provided (both biology and computer science)

Sample Material

£110.00

Add to Basket

Hardback

Published: 24 December 2009

692 Pages | 95 illustration

246x171mm

ISBN: 9780199556670

Bookseller Code (AB)

Handbook Chapters (research areas)

1: An introduction to and an overview of membrane computing, *Gh. Păun & G. Rozenberg*
2: Cell biology for membrane computing, *D. Besozzi & I.I. Ardelean*
3: Computability elements for membrane computing, *Gh. Păun, G. Rozenberg & A. Salomaa*
4: **Catalytic P systems**, *R. Freund, O.H. Ibarra, A. Păun, P. Sosík, & H.-C. Yen*
5: **Communication P systems**, *R. Freund, A. Alhazov, Y. Rogozhin, & S. Verlan*
6: **P automata**, *E. Csuhaj-Varjú, M. Oswald, & G. Vaszil*
7: P systems with string objects, *C. Ferretti, G. Mauri, & C. Zandron*
8: Splicing P systems, *S. Verlan & P. Frisco*

Handbook Chapters

9: **Tissue and population P systems**, *F. Bernardini & M. Gheorghe*
10: Conformon P systems, *P. Frisco*
11: **Active membranes,** *Gh. Păun*
12: **Complexity - Membrane division, membrane creation**, *M.J. Pérez-Jiménez, A. Riscos-Núñez, Á. Romero-Jiménez, & D. Woods*
13: **Spiking neural P systems**, *O.H. Ibarra, A. Leporati, A. Păun, & S. Woodworth*
14: P systems with objects on membranes, *M. Cavaliere, S.N. Krishna, A. Păun, & Gh. Păun*
15: **Petri nets and membrane computing**, *J. Kleijn & M. Koutny*
16: **Semantics of P systems**, *G. Ciobanu*

Handbook Contents

17: Software for P systems, *D. Díaz-Pernil, C. Graciani, M.A. Gutiérrez-Naranjo, I. Pérez-Hurtado, & M.J. Pérez-Jiménez*
18: Probabilistic/stochastic models, *P. Cazzaniga, M. Gheorghe, N. Krasnogor, G. Mauri, D. Pescini, & F.J. Romero-Campero*
19: **Fundamentals of metabolic P systems**, *V. Manca*
20: Metabolic P dynamics, *V. Manca*
21: **Membrane algorithms,** *T.Y. Nishida, T. Shiotani, & Y. Takahashi*
22: **Membrane computing and computer science**, *R. Ceterchi & D. Sburlan*
23: Other developments (**P Colonies***, A. Kelemenová;* **Numerical P systems,** *Gh. Paun, G. Rozenberg*)

Research Frontiers (2013)

M. Gheorghe, Gh. Paun, M.-J. Pérez-Jiménez, G. Rozenberg (eds.):

Research Frontiers of Membrane Computing: Open Problems and Research Topics. Int. J. Found. Comput. Sci. 24(5): 547-624 (2013)

- **Contents**
- 1. A Glimpse to Membrane Computing (The Editors)
- 2. Some General Issues (J. Beal)
- 3. **The Power of Small Numbers** (A. Alhazov)
- 4**. Polymorphic P Systems** (S. Ivanov, A. Alhazov, Y. Rogozhin)
- 5. P Colonies and **dP Automata** (E. Csuhaj-Varjú)
- 6. **Spiking Neural P Systems** (L. Pan, T. Song)

Research Frontiers - continued

- 7. Control Words Associated with P Systems (K. Krithivasan, Gh. Paun, A. Ramanujan)
- 8. **Speeding Up P Automata** (Gy. Vaszil)
- 9. **Space Complexity and the Power of Elementary Membrane Division** (A. Leporati, G. Mauri, A.E. Porreca, C. Zandron)
- 10. The **P-Conjecture and Hierarchies** (N. Murphy)
- 11. **Seeking Sharper Frontiers of Efficiency in Tissue P Systems** (M.J. Pérez-Jiménez, A. Riscos-Núnez, M. Rius-Font, A. Romero-Jiménez)
- 12. **Time-Free Solutions to Hard Computational Problems** (M. Cavaliere)

Research Frontiers - continued

- 13. **Fypercomputations** (Gh. Paun)
- 14. **Numerical P Systems** (C. Vasile, A.B. Pavel, I. Dumitrache, Gh. Paun)
- 15. P Systems: **Formal Verification and Testing** (F. Ipate, M. Gheorghe)
- 16. **Causality, Semantics, Behavior** (O. Agrigoroaiei, B. Aman, G. Ciobanu)
- 17. **Kernel P Systems** (M. Gheorghe)
- 18. **Bridging P and R** (Gh. Paun)
- 19. **P Systems and Evolutionary Computing Interactions**
- (G. Zhang)
- 20. Metabolic P Systems (V. Manca)

Research Frontiers - continued

- 21. **Unraveling Oscillating Structures by Means of P Systems** (T. Hinze)
- 22. Simulating Cells Using P Systems (A. Paun)
- 23. P Systems for Computational Systems and Synthetic Biology (M. Gheorghe,V. Manca, F.J. Romero-Campero)
- 24. Biologically Plausible Applications of Spiking Neural P Systems for an Explanation of Brain Cognitive Functions (A. Obtulowicz)
- 25. Computer Vision (D. Diaz-Pernil, M.A. Gutierrez-Naranjo)
- 26. **Open Problems on Simulation of Membrane Computing Models** (M. Garcıa-Quismondo, L.F. Macıas-Ramos. M.A. Martinez-del-Amor, I. Pérez-Hurtado, L. Valencia-Cabrera)

Further Research Developments

- **Networks of Cells** (R. Freund, S. Verlan et al.)
- **New variants of P colonies** (L. Ciencialová et al., Gy. Vaszil et al.)
- **cP systems and their developments** (R. Nicolescu et al.)
- **New hypercomputing models** (R. Feund et al.),
- and other areas…

Membrane Computing (MC): impacts on Theoretical Computer Science TCS)

P systems theory contributes **to better understand the nature** of (classic) **computational models, algorithms, computing**

Computational completeness, universality:

- P system variants contribute to deeper understand the concept of register machines, two-counter automata, standard Turing machines,
- the role of their descriptional parameters,
- the conditions for computational completeness and universality.

Membrane Computing (MC): impacts on Theoretical Computer Science TCS)

P systems theory contributes **to better understand the nature** of (classic) **computational models, algorithms, computing**

Characteristics of computational models:

- P system variants help in understanding the nature of different variants of counter machines (partially blind counter machines, etc.),
- cellular automata,
- network architectures

Membrane Computing (MC): impacts on Theoretical Computer Science TCS)

P systems theory contributes **to better understand the nature** of (classic) **computational models, algorithms, computing**

What does computing mean?

Variants of functioning of different types of P systems contribute to understand the concept of computing,

- what are the elementary actions,
- what are the conditions of execution of an action,
- what distribution and parallelism mean.

Membrane Computing (MC): impacts on Theoretical Computer Science TCS)

P systems theory contributes **to better understand the nature** of (classic) **computational models, algorithms, computing**

Algorithms in terms of P systems have been elaborated for solving well-known problems, utilizing characteristics of membrane computing.

(Sorting, broadcast, leader election, Byzantine agreement, etc.)

Membrane Computing (MC): impacts on Theoretical Computer Science TCS)

Utilizing characteristics of P systems new approaches, fields have been emerged, for example:

- Membrane algorithms,
- P automata

Membrane algorithms were introduced as a new type of approximate algorithms applying P systems theory to evolutionary computing.

P automata combine properties of classical automata and P systems, thus contribute to better understand the concept of an automaton.

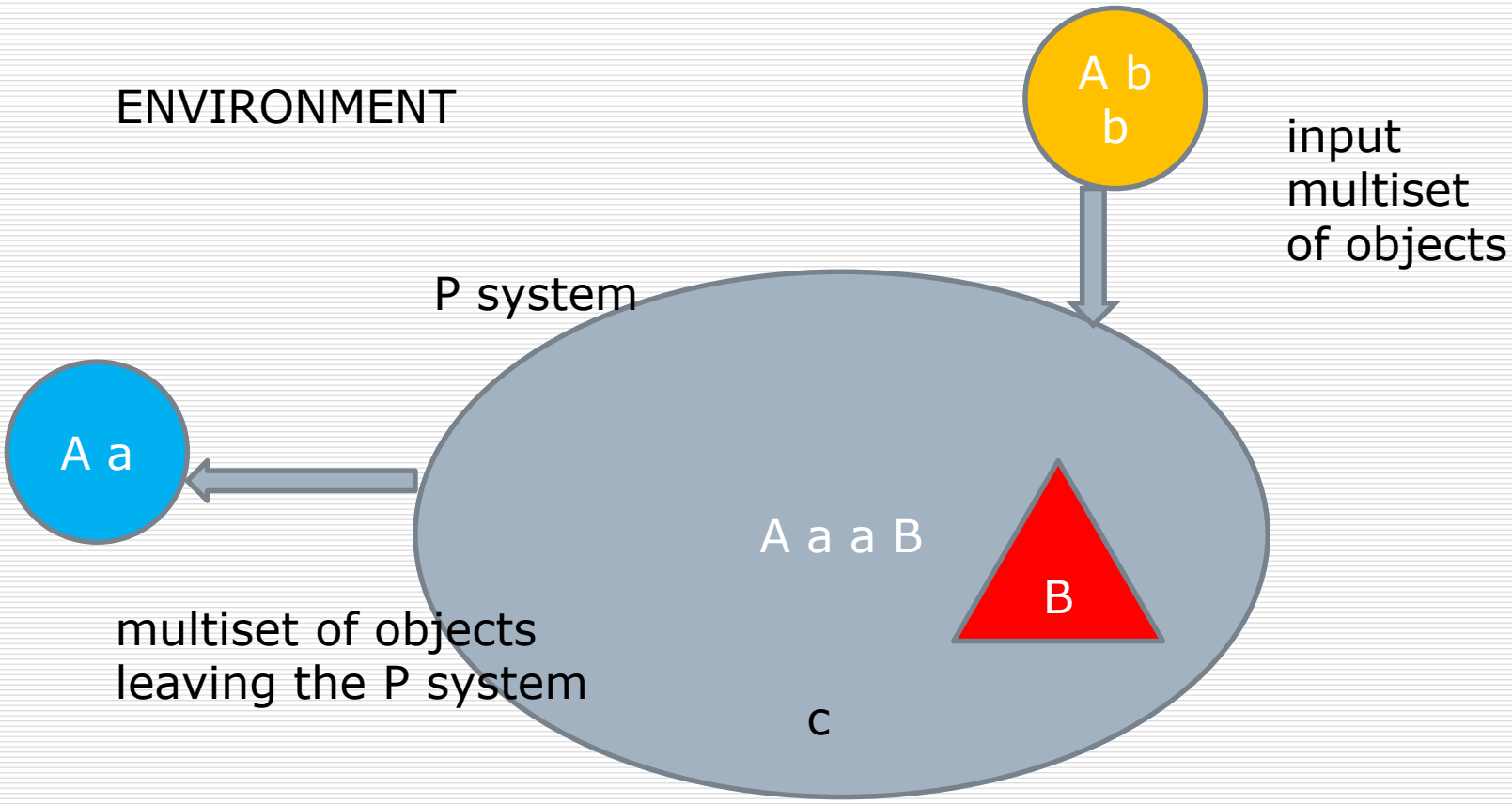
P Automata

- P sytems **communicating** with their own **environments** through multisets of **objects** (input/output multisets of objects);

- their inputs (input sequences) are distinguished as **accepted/not accepted inputs (input sequences)**

- Variants of **accepting P systems** which combine characteristics of **conventional automata** and **P systems which are in interaction with their environments**.

P Automaton


ENVIRONMENT
A b
b
input
multiset
of objects
P system
A a
A a a B
B
multiset of objects
leaving the P system
c

Communication Rules

- $(x, in)|_y$, $(x, out)|_y$ - symport
- $(x, in)|_{\bar{y}}$, $(x, out)|_{\bar{y}}$ - symport
- $(x, out; y, in)|_z$ - antiport
- $(x, out; y, in)|_{\bar{z}}$ - antiport

Symport/antiport rules with **promoters** and **inhibitors**

The rules are applied in **maximally parallel** or **sequential** manner.

P Automaton

$$\Pi = (O, \mu, P_1, \ldots, P_k, c_0, \mathcal{F}),\ k \geq 1,$$

with

- **object** alphabet O,
- **membrane structure** μ,
- **sets of antiport rules** (possibly with promoters) P_i, $1 \leq i \leq k$,
- **initial configuration** $c_0 = (\mu, w_1, \ldots, w_k)$ where $w_i \in O^{(*)}$, $1 \leq i \leq k$, is the initial contents of the ith region,
- and set of **accepting configurations** $\mathcal{F}$ of the form $(\mu, v_1, \ldots, v_k)$, $v_i \in O^{(*)}$, $1 \leq i \leq k$.

Functioning of a Standard P Automaton

The **configurations** of the P automaton are **changed** by applying the rules in the **non-deterministic maximally parallel manner**.

This way, there is a **sequence of multisets** which **enter the system from the environment** during the steps of its computations.

If the computation **ends in an accepting configuration** from $\mathcal{F}$, then this multiset sequence is called an **accepted multiset sequence**.

(accepting by final states (by halting))

P Automaton – Accepted Input Sequences

The **behavior** of the underlying antiport **P system**

can be **described** both

- by the **sequences of multisets of objects** entering the skin region **from the environment** and

- their maps to **words over some alphabets**.

Language of a P Automaton

The **language** accepted by the P automaton Π with respect to a mapping f which maps the multisets from $O^{(*)}$ to finite subsets of Σ^*, is the f-image of the set of multiset sequences accepted by Π.

The **choice of** f **essentially influences** the power of the **P automaton**.

In [Freund, Oswald, 2002] the authors consider a P automata variant where O is mapped by f to the set of strings which **consists of all permutations of the elements of the multiset**.

In the following, we will denote this **mapping by** f_{perm}.

P Automaton, Language

$\mathcal{C}$ is a class of non-erasing linear space computable mappings

Theorem 4.1 *1.* $\mathcal{L}_{seq,\mathcal{C}}(PA) = \mathcal{L}(restricted - 1LOG) \subset \mathcal{L}(1LOG)$ *and*
2. $\mathcal{L}_{maxpar,\mathcal{C}}(PA) = \mathcal{L}(restricted - 1LIN) = \mathcal{L}(1LIN) = \mathcal{L}(CS)$.

If we consider **(possibly erasing) linear space computable** mappings, then **computational completeness** can be obtained.

[Csuhaj-Varjú, Ibarra, Vaszil, 2004])

P Automaton, Language

Theorem 4.3 $\mathcal{L}_{X,f_{perm}}(PA) \subset \mathcal{L}(restricted-1LOG)$ *where* $X \in \{seq, maxpar\}$.

$$\mathcal{L}(\text{RCMA}) = \mathcal{L}(restricted-1LOG)$$

A *special restricted k-counter machine acceptor* (an SRCMA) is a restricted k-counter machine acceptor $M = (Q, \Sigma, k, \delta, q_0, F)$ where δ is defined in such a way that if the length of the string x read in one computational step is l, then the sum of the values stored in the counters can only increase at most as much as $l-1$ in the same computational step.

[Csuhaj-Varjú, Vaszil, 2013]

P Automata, Special Features

The **transitions** are determined by the **actual configuration** of the system, which property resembles a **feature of natural systems**:

the **behavior** of the system is **determined** by its **existing constituents** and their **interaction** with the environment, **there is no abstract component** (workspace) for **influencing the functioning** of the system.

P Automata versus Classical Automata

P automata are different from classical automata since

the **whole input** sequence is not given in advance,
but **will be available step-by-step**, and the
input will also be part of the machine, that is,

the **computing device** and the **input are not separated**.

P automata – Unbounded Features

P automata are tools for **describing languages over infinite alphabets without any extension or additional component** added to the construct (*maximally parallel rule application*)

P finite automata, (Dassow, Vaszil, 2006)

P Automata with Infinite Run

The **functioning** of P automata can be **infinite** or **halting**.

Accepting by final states, **P automata** can be used for describing (possibly) **infinite runs** (sequences of configurations) as well.

P automata can **describe interaction processes not limited in time**.

(ω- P automata, (Freund, Oswald, Staiger, 2003, 2004))

(Red-Green P automata, (Aman, Csuhaj-Varjú, Freund, 2014))

Conclusion and Open Problems

Membrane Computing (may have) has **impact** on the development of **Theoretical Computer Sience**.

Possible new research directions expected in utilizing further the fact that parts of **P systems' theory** corresponds to the **theory of multiset rewriting systems**.

For example, further **connections** between **membrane computing** and **data science** are expected to be explored.