Computing Efficiency in Membrane Systems

Claudio Zandron
claudio.zandron@unimib.it

DISCo - Universita' di Milano-Bicocca, Italy

Future Computing Conference 2021

DIPARTIMENTO DI
INFORMATICA, SISTEMISTICA E
COMUNICAZIONE

Claudio Zandron

Claudio Zandron got the PhD in Computer Science from the University of Milan in 2002.
Since 2006 he is Associate Professor at the Department of Informatics, Systems and Communication of the University of Milano-Bicocca, Italy.
His research interests concern the areas of formal languages, molecular computing models, DNA computing, Membrane Computing and Computational Complexity.

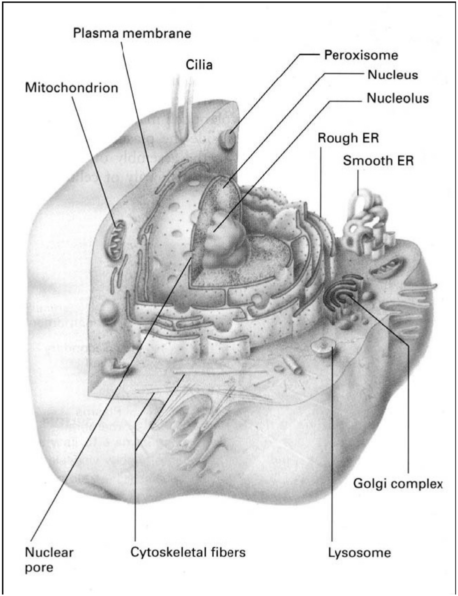
Summary

- Membrane Systems: ideas and definitions
- Membrane Systems with Active Membranes
- Computing Power of Membrane Systems
- Attacking Computationally Hard Problems
- Space Complexity in Membrane Systems with Active Membranes

Membrane Systems: a Bio-inspired computing model

- G. Paun, 1998 (P Systems): computational model inspired from the structure and functioning of the cell
  - Discrete
  - Non-deterministic
  - Maximally parallel application of the rules
- Main components:
  - Cellular structure
  - Chemical substances
  - Cellular reactions
  - Communication of substances

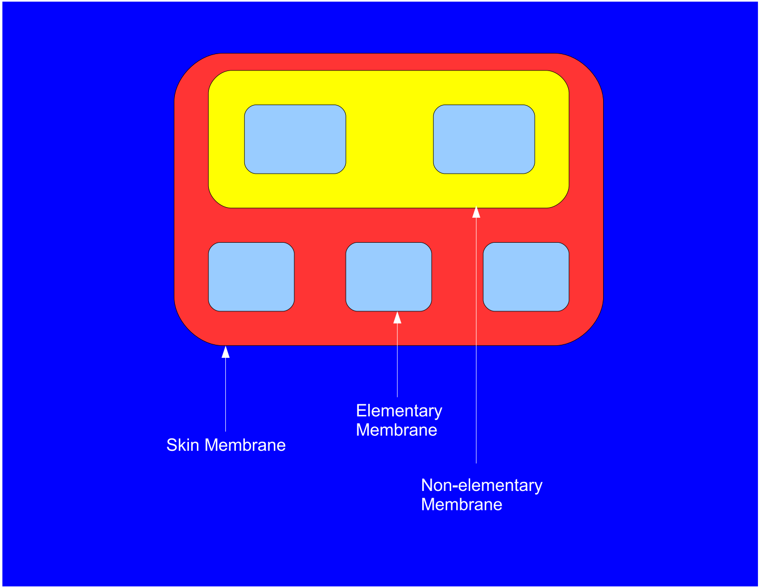
Cell Structure



Membrane structure

- Each membrane defines a REGION (compartment) in the membrane structure
- The most external membrane separates the system and the environment. It is called SKIN
- Some substances are communicated through the membranes
- A membrane is identified by means of a label

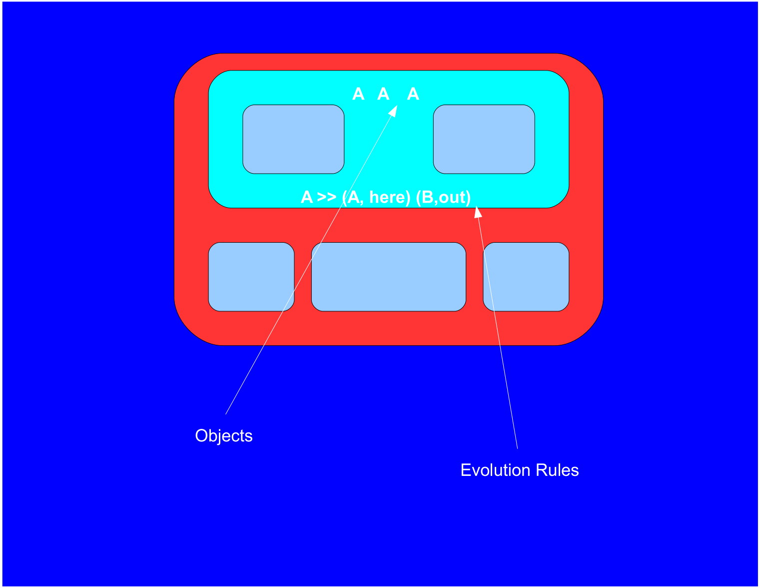
Membrane Structure



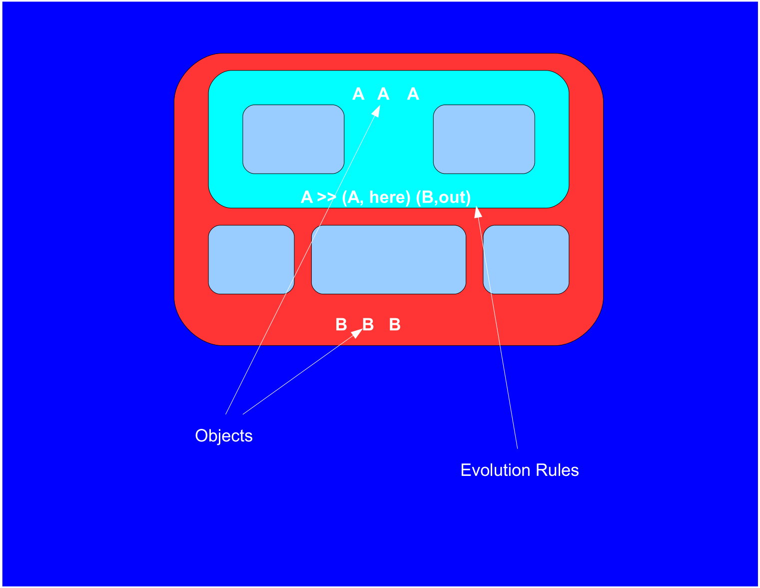
Membrane Systems: Chemicals and Reactions

- Chemicals - Ions, molecules, proteins: multisets of symbols over an alphabet
  - Multiset: each symbol can be present in one or more copies in a region
  - E.g. a^5, b^3, c: five copies of chemical a, three of b, and one of c are present in a region
- A reaction is described by a CF rewriting rule and target indication
  - Chemical on left replaced by chemicals on right
  - Obtained chemicals communicated according to target indication
  - Special Symbol δ: membrane is dissolved
  - E.g. $a \rightarrow (x, here)(y, out)(z, in_3)\delta$

Membrane System



Membrane System



Definition: Membrane System

$$\Pi = (V, \mu, M_1, \ldots, M_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), i_0)$$

- V: Alphabet
- μ: Membrane structure (Ex. $[\ [\ ]_2\ [\ ]_3\ [\ [\ ]_5\ [\ ]_6\ ]_4\ ]_1$)
- M_i: Multisets of symbols (or strings) in V
- R_i: Finite sets of evolution rules $x \to y$, $x \in V^*$, $y = y'$ or $y = y'\delta$ where y' is a string over $(V \times tar)$, $tar \in \{here, out, in_j\}$
- ρ_i: Partial order relations over R_i
- i_0: Output Membrane. If empty, then the output region is the environment

Evolution

- $M_1, \ldots, M_n$: initial configuration
- Rules are applied following the given priorities
- Rules are applied in a non-deterministic way
- All objects evolve in parallel
- All regions evolve in parallel
- Rules can move objects through membranes
  - *here*: the object is not moved
  - *out*: the object is sent to the adjacent external region
  - in_j: the object is sent to the inner membrane with label j

Computation

- **Computation**: Sequence of transitions between two configurations (by means of rules). A computation halts when no further rule can be applied
- **Output**:
  - Objects in i_0 (or outside the skin) when the computation halts
  - $\emptyset$ if the computation never stops

Active Membranes: active role in the computation

- Polarizations: electrical charges (positive +, negative -, or neutral 0) are associated with the membranes.
- Rules are applied according to polarizations
- Membranes can be dissolved
- New membranes can be created by division of existing ones. Objects in the divided membrane are duplicated:
  - Division for elementary membranes
    $[_h \; A \;]_h^{\alpha} \rightarrow [_h \; B \;]_h^{\beta} \;\; [_h \; C \;]_h^{\gamma}$
  - Division for non-elementary membranes
    $[_{h_0} \; [_{h_1} \;]_{h_1}^{+} \ldots [_{h_k} \;]_{h_k}^{+} \; [_{h_{k+1}} \;]_{h_{k+1}}^{-} \ldots [_{h_n} \;]_{h_n}^{-} \;]_{h_0}^{\alpha} \rightarrow$
    $[_{h_0} \; [_{h_1} \;]_{h_1}^{+} \ldots [_{h_k} \;]_{h_k}^{+} \;]_{h_0}^{\beta} \;\; [_{h_0} \; [_{h_{k+1}} \;]_{h_{k+1}}^{-} \ldots [_{h_n} \;]_{h_n}^{-} \;]_{h_0}^{\gamma}$

Definition: Membrane Systems with Active Membranes

$$\Pi = (V, H, \mu, M_1, \ldots, M_n, R)$$

- V: Alphabet
- H: set of labels for membranes
- μ: Membrane structure (Ex. $[\,[\,]_2\ [\,]_3\ [\,[\,]_5\ [\,]_6\ ]_4\ ]_1$)
- M_i: String over V, initial multiset of symbols in region i
- R: Finite sets of evolution rules
- Membranes are marked using polarization: $\{+, -, 0\}$

Developmental Rules

Assume $a \in V, w \in V^*, h \in H, \alpha_i \in \{+, -, 0\}$

- **Object evolution:** $[a \to w]_h^{\alpha_1}$
- **IN communication:** $a\,[\;]_h^{\alpha_1} \to [b]_h^{\alpha_2}$
- **OUT communication:** $[a]_h^{\alpha_1} \to [\;]_h^{\alpha_2}\, b$
- **Dissolution:** $[a]_h^{\alpha_1} \to b$

Division Rules

- **Elementary division:**
  $[a]_h^{\alpha_1} \rightarrow [b]_h^{\alpha_2}\,[c]_h^{\alpha_3}$
- **Non-elementary division:**
  $\left[[\,]_{h_1}^{+} \cdots [\,]_{h_k}^{+} [\,]_{h_{k+1}}^{-} \cdots [\,]_{h_n}^{-}\right]_h^{\alpha_1} \rightarrow$
  $\left[[\,]_{h_1}^{+} \cdots [\,]_{h_k}^{+}\right]_h^{\alpha_2}$
  $\left[[\,]_{h_{k+1}}^{-} \cdots [\,]_{h_n}^{-}\right]_h^{\alpha_3}$
- Non-elementary division: Membranes with neutral polarization are duplicated

Application of the rules

- Maximal parallel semantics
- At each step, each object and membrane can be the subject of only one rule
- If two conflicting rules can be applied: non–deterministic choice
- When a membrane divides, its content is replicated unchanged in the new copy
- OUTPUT: Symbols that exit from the skin in a halting computation

Computing Power of Membrane Systems

- Systems using a single membrane can only generate length sets of context–free languages
- Computing power cannot be extended by using an unlimited number of membranes
- Allowing the dissolution of membranes increases computing power, when at least two membranes are used; universality is not reached.
- To obtain universal systems, further features must be considered: cooperative (non context-free) rules, priorities defining the order of rules application, or structured objects
- Membranes are necessary to reach universality (one membrane does not suffice)

Exploiting Elementary Membrane Division: SAT problem

- SAT - Satisfiability for boolean formulas: a boolean formula Φ in CNF, with
  - n boolean variables $x_1, x_2, \ldots x_n$
  - m clauses
- Question: is there a truth assignment for $x_1, x_2, \ldots x_n$ such that Φ is true?
- Brute force algorithm requires exponential time
- SAT is NP–complete

Solving SAT in linear time

- $[[z_1 \quad a_1 a_2 \ldots a_n]_2^0]_1^0$

Solving SAT in linear time

- $[[z_1 \;\; a_1 a_2 \ldots a_n]_2^0]_1^0$
- $[[z_2 \;\; T_1 \;\; a_2 \ldots a_n]_2^0 \;\; [z_2 \;\; F_1 \;\; a_2 \ldots a_n]_2^0]_1^0$

Solving SAT in linear time

- $[[z_1 \;\; a_1 a_2 \ldots a_n]_2^0]_1^0$
- $[[z_2 \;\; T_1 \;\; a_2 \ldots a_n]_2^0 \;\; [z_2 \;\; F_1 \;\; a_2 \ldots a_n]_2^0]_1^0$
- $[[z_3 \;\; T_1 T_2 \;\; a_3 \ldots a_n]_2^0 \;\; [z_3 \;\; T_1 F_2 \;\; a_3 \ldots a_n]_2^0$
  $[z_3 \;\; F_1 T_2 \;\; a_3 \ldots a_n]_2^0 \;\; [z_3 \;\; F_1 F_2 \;\; a_3 \ldots a_n]_2^0]_1^0$

Solving SAT in linear time

- $[[z_1 \;\; a_1 a_2 \dots a_n]_2^0]_1^0$
- $[[z_2 \;\; T_1 \;\; a_2 \dots a_n]_2^0 \;\; [z_2 \;\; F_1 \;\; a_2 \dots a_n]_2^0]_1^0$
- $[[z_3 \;\; T_1 T_2 \;\; a_3 \dots a_n]_2^0 \;\; [z_3 \;\; T_1 F_2 \;\; a_3 \dots a_n]_2^0$
  $[z_3 \;\; F_1 T_2 \;\; a_3 \dots a_n]_2^0 \;\; [z_3 \;\; F_1 F_2 \;\; a_3 \dots a_n]_2^0]_1^0$
- $\dots$
- $[[z_n \;\; T_1 T_2 \dots T_n]_2^0 \;\; [z_n \;\; T_1 T_2 \dots T_{n-1} F_n]_2^0$
  $\dots \;\; [z_n \;\; F_1 F_2 \dots F_n]_2^0]_1^0$
- In n steps we generate all possible truth assignments

Solving SAT in linear time

- In one step we change the polarization of the membranes using z_n
- $[[T_1\,T_2\ldots T_n]_2^+\;\;[T_1\,T_2\ldots T_{n-1}F_n]_2^+$
  $\ldots\;\;[F_1F_2\ldots F_n]_2^+]_1^0$

Solving SAT in linear time

- In one step we change the polarization of the membranes using z_n
- $[[T_1 T_2 \ldots T_n]_2^+ \ [T_1 T_2 \ldots T_{n-1} F_n]_2^+$
  $\ldots \ [F_1 F_2 \ldots F_n]_2^+]_1^0$
- In one step every symbol T_i (resp. F_i) is replaced by some symbols R_{h_i}
- $1 \leq h_i \leq m$ is the index of a clause satisfied by setting $x_i =$TRUE (resp. $x_i =$FALSE)
- We obtain, for example,
  $[[R_1 R_3 R_1 R_4 \ldots R_6]_2^+ \ [R_7 R_3 R_2 R_3 \ldots R_2]_2^+$
  $\ldots \ [R_2 R_5 R_1 R_5 \ldots R_1]_2^+]_1^0$

Solving SAT in linear time

- In $2m$ steps we check whether or not a membrane contains all R_j, where $1 \leq j \leq m$
- $[[\ldots]_2^- \;\; [\ldots]_2^+ \;\; \ldots \;\; [\ldots]_2^- T \;\; T]_1^0$

Solving SAT in linear time

- In $2m$ steps we check whether or not a membrane contains all R_j, where $1 \leq j \leq m$
- $[[...]_2^- \ [...]_2^+ \ \dots \ [...]_2^- T \ \ T]_1^0$
- After $n + 2m + 2$ steps, eventually a symbol T appears in the skin membrane
- $[[...]_2^- \ [...]_2^+ \ \dots \ [...]_2^- T]_1^- \ \ YES$
- If after exactly $n + 2m + 3$ computation steps we obtain a T, then a symbol YES is sent out through the skin; otherwise a symbol NO is sent out.

Features of the Solution

- Requires linear time
- Requires exponential space
- The solution proposed is said to be **Semiuniform**:
  - Every input instance requires a specific membrane system to be computed
  - Given an input instance x of length n, the membrane system used to solve it can be generated by a deterministic Turing machine in polynomial time w.r.t. n
- The solution is said to be CONFLUENT

Determinism vs Non–determinism

- A Membrane System Π is said to be **deterministic** if there is at most one possible transition from a configuration to the following one, for all possible configurations
- A **non–deterministic** Membrane system Π is said to be **confluent** if the computations of Π are either all accepting or all rejecting. Such a system *accepts* in the former case and *rejects* in the latter
- When not all computations necessarily agree on the result, the system is called **non-confluent**. Non-confluent systems are said to *accept* when there exists an accepting computation, and to *reject* otherwise

Complexity classes for **confluent** Membrane systems

$(N)PMC_T$: languages decided **IN POLYNOMIAL TIME** by (non–)confluent Membrane systems in the class T

- $T = \mathcal{A}M$: systems with both division for elementary and non–elementary membranes
- $T = \mathcal{E}AM$: systems with division for elementary membranes only
- $T = \mathcal{N}AM$: systems without membrane division

Basic properties

- $PMC_T \subseteq NPMC_T$
- $PMC_{\mathcal{N}AM} \subseteq PMC_{\mathcal{E}AM} \subseteq PMC_{\mathcal{A}M}$
- $NPMC_{\mathcal{N}AM} \subseteq NPMC_{\mathcal{E}AM} \subseteq NPMC_{\mathcal{A}M}$

CONFLUENT P systems without division rules

- $P \subseteq PMC_{\mathcal{N}AM}$
  - "Trick": the DTM deciding $L \in P$ is used to solve *DIRECTLY* the problem in polynomial time
  - Then, we build a P system with a single membrane containing either an object *YES*, whenever an input $x \in L$ is given, or *NO*, otherwise. This requires polynomial time.
  - The P system send out the object in a single step

CONFLUENT P systems without division rules

The opposite is also true:

- $PMC_{\mathcal{NAM}} \subseteq P$
  - Idea: simulation of a generic P system Π without membrane division using a DTM M, with a polynomial slowdown
  - We keep track of the NUMBER OF OCCURRENCES of each symbol in each membrane
  - The application of a rule in Π can be simulated by modifying the counters used in M

CONFLUENT systems with elementary division rules only

- We have already seen that SAT is solvable by a family of Membrane systems that make use only of elementary membrane division. It follows: $NP \subseteq PMC_{\mathcal{E}AM}$
- SQRT-3SAT (PP–complete problem) can also be solved by such systems. Hence: $PP \subseteq PMC_{\mathcal{E}AM}$
- Confluent P systems with elementary membrane division can be simulated by Deterministic Turing machines using polinomial space: $PMC_{\mathcal{E}AM} \subseteq PSPACE$

CONFLUENT systems with both types of division rules

- $PMC_{\mathcal{A}M} \subseteq PSPACE$: can be simulated by DTM in polynomial space
- Quantified SAT (QSAT) - SAT using quantifiers: consider a Boolean expression Φ in CNF. Question: $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \dots Q_n x_n \Phi$?
  - QSAT is PSPACE–complete
  - QSAT $\in PMC_{\mathcal{A}M}$
  - $PSPACE \subseteq PMC_{\mathcal{A}M}$
- $PSPACE = PMC_{\mathcal{A}M}$
- What if we remove dissolving action and polarizations? $P = PMC_{\mathcal{A}M}(n\delta, nPol)$!!!

Introducing space complexity classes

- Idea: both objects and membrane need physical space
- Let C_i be a configuration of a P system Π
- *size* $size(C_i)$ is the sum of number of membranes in μ and the total number of objects they contain
- The space required by a halting computation $C = (C_0, C_1, \ldots, C_m)$ of Π is $size(C) = max\{size(C_0), \ldots, size(C_m)\}$
- The *space required by* Π itself is $size(\Pi) =$
  $= max\{size(C) : C \text{ is a halting computation of } \Pi\}$

Some basic result concerning space complexity classes

From results concerning time complexity, it follows immediately:

- $P \subseteq MCSPACE_{\mathcal{N}AM}(O(1))$
- $NP \cup co-NP \subseteq EXPMCSPACE_{\mathcal{E}AM}$
- $PSPACE \subseteq EXPMCSPACE_{\mathcal{A}M}$

Space Complexity Results

- **PSPACE**–complete problem Quantified–3SAT can be solved by Membrane–systems with active membranes using a polynomial amount of space
- Membrane–systems with active membranes using a polynomial amount of space can be simulated by Turing machines using polynomial space
- Hence, **PSPACE** $= PMCSPACE_{\mathcal{AM}}$
- Similarly, **EXPSPACE** $= EXPMCSPACE_{\mathcal{AM}}$
- What about sublinear space?

Sublinear Space Membrane Systems

- Two distinct alphabets: *INPUT* alphabet and *WORK* alphabet
- Input objects cannot be rewritten and do not contribute to the size of a configuration
- Size of a configuration: number of membranes + total number of working objects
- Weaker uniformity condition: *DLOGTIME*-uniformity (*DLOGTIME* Turing machines)

Power of Sublinear Space Membrane Systems

- Idea: compare with logarithmic space Turing machines (or other equivalent models)
- Two problems if we use "standard" techniques:
  - Need for a polynomial number of working objects (violates log–space condition)
  - Need for a polynomial number of rewriting rules (violates uniformity condition)
- Solution: use polarization both to communicate objects and store information

Power of Sublinear Space Membrane Systems

Each Log–space DTM M can be simulated by a *DLOGTIME*-uniform family Π of Membrane systems with active membranes in logarithmic space having:

- A state object $q_{i,w}$: M is in state q, input–head on i-th symbol, work–head on w-th symbol
- $O(\log(n))$ nested membranes (INPUT tape membranes) containing, in the innermost one, the input symbols of M
- $O(\log(n))$ membranes to store the work tape of M (WORK tape membranes).
- Two sets of membranes, which size depends on the dimensions of the input and the working alphabets of M (SYMBOL membranes).

Power of Sublinear Space Membrane Systems

To simulate a computation step of M

- The state object enters the INPUT membranes: the bits corresponding to the actual position of the INPUT head of M are **stored in the polarizations** of the INPUT membranes
- Only the object corresponding to the INPUT symbol actually read can reach the skin
- The state object identifies the symbol actually under the WORK head
- The transition of M can be simulated using the SYMBOLS membranes

Power of Sublinear Space Membrane Systems

- Only a logarithmic number of objects and membranes are required (besides the input objects)
- The family Π is *DLOGTIME*-uniform
- Thus: **L** (class of problems solved by log–space Turing machines) is contained in the class of problems solved by *DLOGTIME*-uniform, log–space Membrane systems with active membranes.

Main resources

- BOOKS:
  - G. Paun, Membrane Computing - An introduction, Springer-Verlag, Berlin, 2002
  - G. Ciobanu, M.J. Perez-Jimenez, G. Paun (Eds), Applications of Membrane Computing, Springer-Verlag, Berlin 2006
  - P. Frisco, Computing with Cells. Advances in Membrane Computing, Oxford University Press, 2009
  - G. Paun, G. Rozenberg, A. Salomaa (eds.), The Oxford Handbook of Membrane Computing, Oxford University Press, 2010
- INTERNET: P systems web page: http://ppage.psystems.eu