Cell Division versus Membrane Fission: A Computational Complexity Perspective

Mario J. Pérez Jiménez

Research Group on Natural Computing
Dpto. Ciencias de la Computación e Inteligencia Artificial
University of Sevilla
Academia Europaea (The Academy of Europe)

www.cs.us.es/~marper marper@us.es

13th Brainstorming Week on Membrane Computing
Sevilla, Spain, February 2, 2015
Cell division (I)

- Binary fission (prokaryotic cells)
- Mitosis (eukaryotic cells)
- Meiosis (eukaryotic cells)
Cell division (I)

One of the basic processes in the cell life cycle.
Cell division (I)

One of the basic processes in the cell life cycle.

- *Binary fission* (prokaryotic cells)
- *Mitosis* (eukaryotic cells)
- *Meiosis* (eukaryotic cells)
Cell division (II)


Cell division (II)

Cell division inspired mechanism in Membrane Computing:

» P systems with active membranes (membrane division rules)$^1$

- Evolution, Send-in, Send-out, Dissolution, Membrane division rules.
- Computational completeness.
- Computational efficiency (a semi-uniform polynomial time solution for SAT by using communication rules with length bounded by the number of clauses of the input$^2$).


Cell division (II)

Cell division inspired mechanism in Membrane Computing:

- **P systems with active membranes (membrane division rules)**
  - Evolution, Send-in, Send-out, Dissolution, Membrane division rules.
  - Computational completeness.
  - Computational efficiency (a semi-uniform polynomial time solution for SAT by using communication rules with length bounded by the number of clauses of the input).

- **Tissue P systems with cell division (cell division rules)**
  - Symport/antiport, Cell division rules.
  - Computational completeness.
  - Computational efficiency (a uniform polynomial time solution for SAT by using communication rules with length at most 5).

---


Membrane fission (I)

Lipid membranes:

▶ Plasma membrane

Separates the interior of a cell from its environment.

▶ Membrane compartments

Concentrations barriers allowing incorporation of material (donor-acceptor membrane).
Membrane fission (I)

Lipid membranes:

- **Plasma membrane**
  - Separates the interior of a cell from its environment.

- **Membrane compartments**
  - Concentrations barriers allowing incorporate material (donor-acceptor membrane).
Membrane fission (II)


Membrane fission inspired mechanism in Membrane Computing:

▶ P systems with active membranes:
  * Membrane separation rules associated with subsets of the working alphabet\(^4\) (a semi-uniform polynomial time solution to SAT by using evolution rules with length at most 5).
  * Membrane separation rules associated with a prefixed partition of the working alphabet\(^5\) (computational completeness + a uniform polynomial time solution to SAT by using evolution rules with length at most 5).

▶ Tissue P systems with cell separation:
  * Cell separation rules associated with a prefixed partition of the working alphabet\(^6\) (a semi-uniform polynomial time solution to SAT by using evolution rules with length at most 5).

---


Trans-membrane transport

Trans-membrane transport

Networks of membranes which compute by communication only:

- Symport/antiport rules\(^7\).
- Used both for communication with the environment and for direct communication between membranes.
- The environment plays an active role.
- Computational completeness.

---

Cell-like P systems with symport/antiport rules

Computational completeness.

Cell-like versus Tissue-like: Symport/antiport rules:

- **Cell-like**
  - Set of rules: Each membrane has associated a set of rules
  - Structure: Rooted tree: defined in an explicit way
  - Environment: Only skin membrane can communicate with it
  - Communication: Two membranes: in an indirect way

- **Tissue-like**
  - Set of rules: Associated with the system
  - Structure: Directed graph: defined by the set of rules
  - Environment: Any cell can communicate with it
  - Communication: Two cells: directly

---

Cell-like P systems with symport/antiport rules

\[ \Pi = (\Gamma, \Sigma, \mu, M_1, \ldots, M_q, R_1, \ldots, R_q, i_{in}, i_{out}) \]

- Computational completeness.

---

Cell-like P systems with symport/antiport rules

\[ \Pi = (\Gamma, \Sigma, \mu, M_1, \ldots, M_q, R_1, \ldots, R_q, i_{in}, i_{out}) \]

* Computational completeness.

Cell-like versus Tissue-like: Symport/antiport rules:

<table>
<thead>
<tr>
<th></th>
<th>Cell-like</th>
<th>tissue-like</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set of rules</strong></td>
<td>Each membrane has associated a set of rules</td>
<td>Associated with the system</td>
</tr>
<tr>
<td><strong>Structure</strong></td>
<td>Rooted tree: defined in an explicit way</td>
<td>Directed graph: defined by the set of rules</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>Only skin membrane can communicate with it</td>
<td>Any cell can communicate with it</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>Two membranes: in an indirect way</td>
<td>Two cells: directly</td>
</tr>
</tbody>
</table>

For each $k \geq 1$, we consider the following classes of recognizer cell-like $P$ systems (set of rules associated with each membrane)

\begin{itemize}
  \item[$\star$] $(u, \text{out}; v, \text{in})$, for $u, v \in \Gamma^*$ (symport-antiport rules) whose length ($|u| + |v|$) is at most $k$.
  \item[$\star$] $[a]_i \to [b]_i[c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$ (division rules).
\end{itemize}
For each $k \geq 1$, we consider the following classes of recognizer cell-like P systems (set of rules associated with each membrane)

**CDC($k$)**

* $(u, \text{out}; v, \text{in})$, for $u, v \in \Gamma^*$ (symport-antiport rules) whose length $(|u| + |v|)$ is at most $k$.
* $[a]_i \rightarrow [b][c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$ (division rules).

**CSC($k$)**

* $(u, \text{out}; v, \text{in})$, for $u, v \in \Gamma^*$ (symport-antiport rules) whose length $(|u| + |v|)$ is at most $k$.
* $[a]_i \rightarrow [\Gamma_1]_i[\Gamma_2]_i$, where $i \in \{1, 2, \ldots, q\}$, $a \in \Gamma$, $i \neq i_{out}$ and $\{\Gamma_1, \Gamma_2\}$ is a fixed partition of $\Gamma$ (separation rules).
For each $k \geq 1$, we consider the following classes of recognizer cell-like P systems (set of rules associated with each membrane)

\[
\text{CDC}(k)
\]

\begin{itemize}
\item $(u, out; v, in)$, for $u, v \in \Gamma^*$ (symport-antiport rules) whose length $(|u| + |v|)$ is at most $k$.
\item $[a]_i \rightarrow [b]_i[c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$ (division rules).
\end{itemize}

\[
\text{CSC}(k)
\]

\begin{itemize}
\item $(u, out; v, in)$, for $u, v \in \Gamma^*$ (symport-antiport rules) whose length $(|u| + |v|)$ is at most $k$.
\item $[a]_i \rightarrow [\Gamma_1]_i[\Gamma_2]_i$, where $i \in \{1, 2, \ldots, q\}$, $a \in \Gamma$, $i \neq i_{out}$ and $\{\Gamma_1, \Gamma_2\}$ is a fixed partition of $\Gamma$ (separation rules).
\end{itemize}

The semantics is similar to the tissue P systems with cell division or separation.
Cell-like P systems with symport/antiport rules and “without environment”

\[ \Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, M_1, \ldots, M_q, R_1, \ldots, R_q, i_{in}, i_{out}) \] such that \( \mathcal{E} = \emptyset \).

- No objects initially located in the environment of the system available in an arbitrary number of copies.
- In such P systems objects in the environment always have finite multiplicity.
Cell-like P systems with symport/antiport rules and "without environment"

\[ \Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, M_1, \ldots, M_q, R_1, \ldots, R_q, i_{\text{in}}, i_{\text{out}}) \text{ such that } \mathcal{E} = \emptyset. \]

- * No objects initially located in the environment of the system available in an arbitrary number of copies.
- * In such P systems objects in the environment always have finite multiplicity.

The classes \( \widehat{\text{CDC}}(k) \) and \( \widehat{\text{CSC}}(k) \)
Recent results

- $P = \text{PMC}_{\text{CDC}(1)} = \text{PMC}_{\text{CSC}(1)}$ (dependency graph technique).

- $\text{NP} \cup \text{co-NP} \subseteq \text{PMC}_{\text{CDC}(2)}$ ($\text{SAT} \in \text{PMC}_{\text{CDC}(3)}$).

- $\text{NP} \cup \text{co-NP} \subseteq \text{PMC}_{\text{CDC}(2)}$ ($\text{HAM} - \text{CYCLE} \in \text{PMC}_{\text{CDC}(2)}$).

- $P = \text{PMC}_{\text{CSC}(2)}$ (algorithmic technique).

- For each $k \geq 1$, $\text{PMC}_{\text{CDC}(k)} = \text{PMC}_{\hat{\text{CDC}}(k)}$ (simulation technique).
Recent results

- $P = \text{PMC}_{CD(1)} = \text{PMC}_{SC(1)}$ (dependency graph technique).

- $P = \text{PMC}_{SC(2)}$ (algorithmic technique).
Recent results

- $P = \text{PMC}_{\text{CDC}(1)} = \text{PMC}_{\text{CSC}(1)}$ (dependency graph technique).

- $P = \text{PMC}_{\text{CSC}(2)}$ (algorithmic technique).

- $\text{NP} \cup \text{co-NP} \subseteq \text{PMC}_{\text{CSC}(3)}$ ($\text{SAT} \in \text{PMC}_{\text{CSC}(3)}$).
Recent results

- $P = \text{PMCC}_{\text{CDC}(1)} = \text{PMCC}_{\text{CSC}(1)}$ (dependency graph technique).
- $P = \text{PMCC}_{\text{CSC}(2)}$ (algorithmic technique).
- $\text{NP} \cup \text{co-NP} \subseteq \text{PMCC}_{\text{CSC}(3)} \ (\text{SAT} \in \text{PMCC}_{\text{CSC}(3)})$.
- $\text{NP} \cup \text{co-NP} \subseteq \text{PMCC}_{\text{CDC}(2)} \ ( \text{HAM} - \text{CYCLE} \in \text{PMCC}_{\text{CDC}(2)})$. 
Recent results

- $P = \text{PMC}_{\text{CD}(1)} = \text{PMC}_{\text{SC}(1)}$ (dependency graph technique).
- $P = \text{PMC}_{\text{SC}(2)}$ (algorithmic technique).
- $\text{NP} \cup \text{co-NP} \subseteq \text{PMC}_{\text{SC}(3)} \quad (\text{SAT} \in \text{PMC}_{\text{SC}(3)})$.
- $\text{NP} \cup \text{co-NP} \subseteq \text{PMC}_{\text{CD}(2)} \quad (\text{HAM} - \text{CYCLE} \in \text{PMC}_{\text{CD}(2)})$.
- $P = \text{PMC}_{\text{SC}}$ (algorithmic technique).
Recent results

- $\mathbf{P} = \mathbf{PMC}_{\mathbf{CDC}(1)} = \mathbf{PMC}_{\mathbf{CSC}(1)}$ (dependency graph technique).

- $\mathbf{P} = \mathbf{PMC}_{\mathbf{CSC}(2)}$ (algorithmic technique).

- $\mathbf{NP} \cup \mathbf{co-NP} \subseteq \mathbf{PMC}_{\mathbf{CSC}(3)} (\text{SAT} \in \mathbf{PMC}_{\mathbf{CSC}(3)})$.

- $\mathbf{NP} \cup \mathbf{co-NP} \subseteq \mathbf{PMC}_{\mathbf{CDC}(2)} (\text{HAM} - \text{CYCLE} \in \mathbf{PMC}_{\mathbf{CDC}(2)})$.

- $\mathbf{P} = \mathbf{PMC}_{\mathbf{CSC}}$ (algorithmic technique).

- For each $k \geq 1$, $\mathbf{PMC}_{\mathbf{CDC}(k)} = \mathbf{PMC}_{\mathbf{CDC}(k)}$ (simulation technique).
Proof techniques

Dependency graph

★ Construction of a directed graph \((dependency graph)\) \(G_{\Pi}\) associated with a P system \(\Pi\) verifying:

- There exists an accepting computation of \(\Pi\) if and only if there exists a path between two distinguished nodes in the dependency graph associated with it.
Proof techniques

Dependency graph

★ Construction of a directed graph (dependency graph) $G_{\Pi}$ associated with a P system $\Pi$ verifying:

• There exists an accepting computation of $\Pi$ if and only if there exists a path between two distinguished nodes in the dependency graph associated with it.

Algorithmic

★ A deterministic algorithm $A$ working in polynomial time that receives as input a P system $\Pi$ and an input multiset $m$ of $\Pi$. Then, algorithm $A$ reproduces the behaviour of a single computation of $\Pi + m$. 
Proof techniques

**Dependency graph**

* Construction of a directed graph (dependency graph) $G_\Pi$ associated with a P system $\Pi$ verifying:
  
  - There exists an accepting computation of $\Pi$ if and only if there exists a path between two distinguished nodes in the dependency graph associated with it.

**Algorithmic**

* A deterministic algorithm $A$ working in polynomial time that receives as input a P system $\Pi$ and an input multiset $m$ of $\Pi$. Then, algorithm $A$ reproduces the behaviour of a single computation of $\Pi + m$.

**Simulation**

* $\Pi'$ simulates $\Pi$ in an efficient way if the following holds:

  - (a) $\Pi'$ can be constructed from $\Pi$ by a DTM working in polynomial time.
  - (b) There exists an injective function, $f$, from $\text{Comp}(\Pi)$ onto $\text{Comp}(\Pi')$ such that:
    - There exists a DTM that constructs $f(C)$ from computation $C$ in polynomial time.
    - A computation $C$ is an accepting computation if and only if $f(C)$ is an accepting one.
    - There exists a polynomial function $p(n)$ verifying $|f(C)| \leq p(|C|)$ for each $C \in \text{Comp}(\Pi)$. 
Frontiers of the efficiency

<table>
<thead>
<tr>
<th>NonEfficiency (Feasible)</th>
<th>Efficiency (Presumably Efficient)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>CD</em>&lt;sub&gt;1&lt;/sub&gt;</td>
<td><em>CD</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td>(length)</td>
</tr>
<tr>
<td><em>CS</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td><em>CS</em>&lt;sub&gt;3&lt;/sub&gt;</td>
<td>(length)</td>
</tr>
<tr>
<td><em>CS</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td><em>CD</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td>(kind)</td>
</tr>
<tr>
<td><em>CS</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td><em>CD</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td>(kind)</td>
</tr>
<tr>
<td><em>CS</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td><em>CS</em>&lt;sub&gt;2&lt;/sub&gt;</td>
<td>(environment)</td>
</tr>
</tbody>
</table>
Frontiers of the efficiency

<table>
<thead>
<tr>
<th>NonEfficiency (Feasible)</th>
<th>Efficiency (Presumably Efficient)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{CDC}(1)$</td>
<td>$\text{CDC}(2)$</td>
<td>(length)</td>
</tr>
<tr>
<td>$\text{CSC}(2)$</td>
<td>$\text{CSC}(3)$</td>
<td>(length)</td>
</tr>
<tr>
<td>$\text{CSC}(2)$</td>
<td>$\text{CDC}(2)$</td>
<td>(kind)</td>
</tr>
<tr>
<td>$\text{CSC}(2)$</td>
<td>$\text{CDC}(2)$</td>
<td>(kind)</td>
</tr>
<tr>
<td>$\text{CSC}$</td>
<td>$\text{CSC}$</td>
<td>(environment)</td>
</tr>
</tbody>
</table>

Each such frontier provides a new way to tackle the P versus NP problem.
Does structure matter?

Similar results for:

- Tissue P systems with symport/antiport rules (cell division/cell separation).
- Cell-like P systems with symport/antiport rules (cell division/cell separation).

Open questions:

1. Cell-like P systems with symport/antiport rules, can be efficiently simulated by tissue P systems with symport/antiport rules?
2. Tissue P systems with symport/antiport rules, can be efficiently simulated by cell-like P systems with symport/antiport rules?

Idea: Complexity aspects on Tissue P systems with active cells.
Does structure matter?

Similar results for:

- Tissue P systems with symport/antiport rules (cell division/cell separation).
- Cell-like P systems with symport/antiport rules (cell division/cell separation).

Open questions:

1. **Cell-like** P systems with symport/antiport rules, can be efficiently simulated by **tissue** P systems with symport/antiport rules?

2. **Tissue** P systems with symport/antiport rules, can be efficiently simulated by **cell-like** P systems with symport/antiport rules?
Does structure matter?

Similar results for:

- Tissue P systems with symport/antiport rules (cell division/cell separation).
- Cell-like P systems with symport/antiport rules (cell division/cell separation).

Open questions:

1. **Cell-like** P systems with symport/antiport rules, can be efficiently simulated by **tissue** P systems with symport/antiport rules?

2. **Tissue** P systems with symport/antiport rules, can be efficiently simulated by **cell-like** P systems with symport/antiport rules?

Idea: Complexity aspects on **Tissue P systems with active cells**.