Interactions in P systems

Artiom Alhazov

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
artiom@math.md

October 17, 2006
Computational Biomodelling Laboratory
Åbo Academi University, Turku

References:
http://psystems.disco.unimib.it
http://www.geocities.com/aartiom/pub_aa.html
Outline

1 General Definitions
   - Multisets
   - Processing
   - Parallel
   - Distributive
1 General Definitions
   - Multisets
   - Processing
   - Parallel
   - Distributive

2 Object-Object Interaction
   - By rewriting: catalysts
   - Bi-stable
   - By communication: protons
   - Communication only
General Definitions

1. Multisets
2. Processing
3. Parallel
4. Distributive

Object-Object Interaction

2. By rewriting: catalysts
3. Bi-stable
4. By communication: protons
5. Communication only

Object-Membrane Interaction

3. Active membranes
4. Properties
5. Polarizations
6. Miscellaneous
Let $O$ be a finite alphabet

- A multiset is a set with multiplicities
- Represented by string $w \in O^*$

Example

```
  a   c   a  
 b   a   b   c 
```

3 copies of $a$, 2 copies of $b$ and 2 copies of $c$
Reaction rules

- \( u \rightarrow v \)
- Consume a multiset \( u \), and
- Produce a multiset \( v \)

**Example**

<table>
<thead>
<tr>
<th>a</th>
<th>c</th>
<th>a</th>
<th>ba → bc</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>
Maximal parallelism; non-determinism

- **Parallelism**
  - The same rule may be applied multiple times
  - Different rules may be applied simultaneously
Maximal parallelism; non-determinism

- **Parallelism**
  - The same rule may be applied multiple times
  - Different rules may be applied simultaneously

- **Maximality**
  - Objects that are not consumed, remain idle
  - No rule should be applicable to the idle objects
Maximal parallelism; non-determinism

- **Parallelism**
  - The same rule may be applied multiple times
  - Different rules may be applied simultaneously

- **Maximality**
  - Objects that are not consumed, remain idle
  - No rule should be applicable to the idle objects

**Example**

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>ba → bc</th>
<th>applied once</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>b</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>ca → a</th>
<th>applied 2 times</th>
</tr>
</thead>
</table>
```
Maximal parallelism; non-determinism

- **Parallelism**
  - The same rule may be applied multiple times
  - Different rules may be applied simultaneously

- **Maximality**
  - Objects that are not consumed, remain idle
  - No rule should be applicable to the idle objects

**Example**

\[
\begin{array}{ccc}
  a & c & a \quad \text{ba} \rightarrow bc \text{ applied 2 times} \\
  b & a & b & c \quad \text{ca} \rightarrow a \text{ applied once}
\end{array}
\]
**Example**

Rules $ba \rightarrow bc$, $ca \rightarrow a$, starting multiset $b^2a^3c^2$.

$bbaaacc \Rightarrow (bc)^2(a)c$:
Example

Rules $ba \rightarrow bc$, $ca \rightarrow a$, starting multiset $b^2a^3c^2$.

$bbaaacc \Rightarrow (bc)^2(a)c$:

$bbaaacc \Rightarrow b(bc)(a)^2$:
Computation

Example

Rules $ba \rightarrow bc$, $ca \rightarrow a$, starting multiset $b^2a^3c^2$.

$bb\!aa\!ac\!c \Rightarrow (bc)^2(a)c$:

- $bb\!ac\!c\!c \Rightarrow bb\!c\!c\!c\!c\!c$.
- $bb\!ac\!c\!c \Rightarrow bb\!ac\!c \Rightarrow bb\!ac \Rightarrow bb\!cc$.
- $bb\!ac\!c\!c \Rightarrow bb\!ac\!c \Rightarrow bb\!ac \Rightarrow bba \Rightarrow bbc$.

$bb\!aa\!ac\!c\!c \Rightarrow b(bc)(a)^2$:
Example

Rules $ba \rightarrow bc$, $ca \rightarrow a$, starting multiset $b^2a^3c^2$.

$bbaaacc \Rightarrow (bc)^2(a)c$:

- $bbaacc \Rightarrow bbccccc$.
- $bbaccc \Rightarrow bbacc \Rightarrow bbcccc$.
- $bbaccc \Rightarrow bbaccc \Rightarrow bbac \Rightarrow bbcc$.
- $bbaccc \Rightarrow bbaccc \Rightarrow bbac \Rightarrow bba \Rightarrow bbcc$.

$bbaaacc \Rightarrow b(bc)(a)^2$:

- $bbaac \Rightarrow bbccc$.
- $bbaac \Rightarrow bbac \Rightarrow bbcc$.
- $bbaac \Rightarrow bbac \Rightarrow bba \Rightarrow bbcc$. 
Objects are in regions
Regions are delimited by membranes
Associated to the region directly inside
External region is called the environment
Structure

- Objects are in regions
- Regions are delimited by membranes
- Associated to the region directly inside
- External region is called the environment

Cell-like systems
- Membranes are nested
- Serve as channels
- Tree structure
Objects are in regions
Regions are delimited by membranes
Associated to the region directly inside
External region is called the environment

- **Cell-like systems**
  - Membranes are nested
  - Serve as channels
  - Tree structure

- **Tissue-like systems**
  - Cells are in the environment
  - Connected by channels
  - Graph structure
Cell-like VS tissue-like

$Env$

Region1  Region2  Region3  Region4
Cell-like VS tissue-like

Env

Region1
Region2
Region3
Region4

Region1 → Region2
Region3 → Env
Region4 → Region5
Cell-like VS tissue-like

Artiom Alhazov
Interactions in P systems
Result

- Objects can be moved between regions
- Transitional model: target indications
- Destinations are specified in the right side of the rules
- Designated output region (env. = 0 or membrane)
Objects can be moved between regions

Transitional model: target indications

Destinations are specified in the right side of the rules

Designated output region (env. = 0 or membrane)

Consider at halting

total number of objects: a number

number of objects of each kind: a vector

order in which objects come: a word
Non-cooperative rules work like a 0L system

Rules of type $u \rightarrow v$, $|u| \leq 2$ are already too powerful.
Two objects by rewriting

- Non-cooperative rules work like a $0L$ system
- Rules of type $u \rightarrow v$, $|u| \leq 2$ are already too powerful.

**Catalysts** $C \subseteq O$

- Catalytic rules are of type $ca \rightarrow cv$, $c \in C$, $a \in O \setminus C$, $v \in (O \setminus C)^*$
- We may assume the catalysts are distinct
- Essentially, a catalyst ensures that out of associated rules $a \rightarrow v$ at most one will be applied.
Catalysts are universal

[Freund et al.]

Theorem

*Purely catalytic P systems with 3 catalysts generate RE.*

- Like 3 CF grammars
- working on the same sentential form
- only interacting by competing for resources
- With non-cooperative rules this can be improved
Catalysts are universal
[Freund et al.]

**Theorem**

*Purely catalytic P systems with 3 catalysts generate RE.*

- Like 3 CF grammars
- working on the same sentential form
- only interacting by competing for resources
- With non-cooperative rules this can be improved

**Theorem**

*P systems with 2 catalysts generate RE.*
Bi-stable catalysts

- \( a \rightarrow u, \)
- \( ca \rightarrow c'u \)

**Theorem**

*P systems with 1 bi-stable catalyst generate RE.*

- Like a 0L system and a CF grammar
- the latter has 1-bit memory
Interaction by moving objects

- **Symport** rules: \((v, out)\) or \((u, in)\)
- **Antiport** rule: \((v, out; u, in)\)
- **weight**: \(\max(|u|, |v|)\)
Interaction by moving objects

- **Symport** rules: \((v, \text{out})\) or \((u, \text{in})\)
- **Antiport** rule: \((v, \text{out}; u, \text{in})\)
- weight: \(\max(|u|, |v|)\)
- Minimal cooperation: rules envolve at most 2 objects
- in one way: \((\text{sym}_1, \text{anti}_1)\) or \((\text{sym}_2)\)
### Interaction by moving objects

- **Symport** rules: \((v, \text{out})\) or \((u, \text{in})\)
- **Antiport** rule: \((v, \text{out}; u, \text{in})\)
- weight: \(\max(|u|, |v|)\)
- Minimal cooperation: rules evolve at most 2 objects
  - in one way: \((\text{sym}_1, \text{anti}_1)\) or \((\text{sym}_2)\)

- Evolution–communication: non-cooperative evolution plus communication with minimal cooperation
A proton

- is one of the interacting objects
- is never mentioned in evolution rules
- is essentially an $m$-stable catalyst of communication

The previous Theorem was proved as a corollary of

Theorem

$P$ systems with 1 proton and 2 membranes generate $RE$
More results

**Time-freeness**: rules are not necessarily executed in 1 step; the result should be independent.

**Theorem**

4 protons and 2 membranes suffice for generating RE
More results

**Time-freeness:** rules are not necessarily executed in 1 step; the result should be independent.

**Theorem**

4 protons and 2 membranes suffice for generating RE

- **Accepting:** the input multiset is placed in a designated region; it is accepted iff the system halts
- **Determinism:** the maximal multiset of applicable rules is always unique

**Theorem**

3 membranes suffice for accepting PsRE
Symport/antiport only
Rules envolving at most 3 objects

- There are infinitely many objects from $E \subseteq O$ in the environment

Known results

Theorem

$P$ systems with $(\text{sym}_1, \text{anti}_{2/1})$ generate NRE and deterministically accept PsRE, already with 1 membrane.

The accepting case also holds for $(\text{sym}_3)$
There are infinitely many objects from $E \subseteq O$ in the environment.

**Known results**

**Theorem**

$P$ systems with $(\text{sym}_1, \text{anti}_{2/1})$ generate NRE and deterministically accept $PsRE$, already with 1 membrane. The accepting case also holds for $(\text{sym}_3)$.

**Improved result:**

**Theorem**

$P$ systems with $(\text{sym}_3)$ and 1 membrane generate at least $N_7RE$.
Minimal cooperation

Rules envolving at most 2 objects: \((sym_1, anti_1)\) or \((sym_2)\)

“Clean” result:

**Theorem**

*Such P systems with 3 membranes generate NRE*
Minimal cooperation
Rules envolving at most 2 objects: \((sym_1, anti_1)\) or \((sym_2)\)

“Clean” result:

**Theorem**

*Such P systems with 3 membranes generate NRE*

Latest and optimal: 1 “garbage” object

**Theorem**

*Such P systems with 2 membranes generate at least \(N_1\) RE; if a set containing 0 is generated, that set is finite.*

**Theorem**

*Such P systems with 1 membrane only generate finite sets.*
“Heavy” symport/antiport rules are considered; information is stored in big multisets over a small alphabet.

**Theorem**

\[ P \text{ systems with } m \geq 2 \text{ membranes and } n \geq 1 \text{ symbols, } m + n \geq 6 \text{ generate NRE} \]
“Heavy” symport/antiport rules are considered; information is stored in big multisets over a small alphabet.

**Theorem**

*P systems with* $m \geq 2$ *membranes and* $n \geq 1$ *symbols, $m + n \geq 6$ generate NRE*

- Generating vectors
- Tissue P systems
- smaller $(m, n)$

are also studied.
Active membranes

\( h \in H \) is a membrane label, \( e, e', e'' \in E \) are polarizations

(a) \([ a \rightarrow v ]^e_h \) evolution

(b) \( a[ ]^e_h \rightarrow [ b ]^e'_h \) send in

(c) \( [ a ]^e_h \rightarrow [ ]^e'_h b \) send out

(d) \( [ a ]^e_h \rightarrow b \) dissolution

(e) \( [ a ]^e_h \rightarrow [ b ]^e'_h [ c ]^e''_h \) division of elementary membrane

etc.

Question: how much information does a membrane need to carry for the systems to be complete or efficient
Properties

- Determinism
- Confluence
Properties

- Determinism
- Confluence
- Uniform solution
- Semi-uniform solution
Properties

- Determinism
- Confluence
- Uniform solution
- Semi-uniform solution
- etc.
Polarizations

Theorem

Systems with 2 polarizations are complete (in 1 membrane) and efficient.

Theorem

Systems with 1 polarization are complete (unbounded membrane division) and efficient (with non-elementary membrane division).
Polarizations

Theorem

Systems with 2 polarizations are complete (in 1 membrane) and efficient.

Theorem

Systems with 1 polarization are complete (unbounded membrane division) and efficient (with non-elementary membrane division).

Minimal parallelism: at least one rule associated to every membrane is applied, if possible.

Theorem

Minimally parallel systems are efficient with 6 or 4 polarizations, depending on the rules used.
- Using separation instead of division
- Changing membrane labels instead of polarizations
- Descriptional complexity parameters
- ...
• Using separation instead of division
• Changing membrane labels instead of polarizations
• Descriptive complexity parameters
• …
• Cooperation by promoters or inhibitors
• Sorting P systems
• Solving graph problems
• …
- Using separation instead of division
- Changing membrane labels instead of polarizations
- Descriptive complexity parameters
- ... 
- Cooperation by promoters or inhibitors
- Sorting P systems
- Solving graph problems
- ... 
- Topics outside the scope of this presentation