# Membrane Systems and Their Relation to the Chemical Programming Paradigm

**Péter Battyányi and György Vaszil**

Department of Computer Science, Faculty of Informatics
University of Debrecen

2015 February

# Outline

1. Membrane computing
2. Chemical programming
3. Results: transformation of a $P$-system into the chemical calculus

# Outline

- Former results: a transformation of cooperative P systems with targetted rules without dissolution into the chemical calculus (in: Describing Membrane Computations with a Chemical Calculus, *Fundamenta Informaticae* (134), pp. 39-50)
- Novelty: extension of the transformation to the case of membrane dissolution, priority rules, catalisators, and promoter/inhibitor sets for rules

# Membrane systems

Membrane computing is a biologically inspired distributed parallel type model of computation.

Unconventional model:

- Its operation is based on manipulation of multisets of objects in compartments defined by the membrane structure.

Nature motivated model:

- The unusual principle is inspired by some natural process or phenomena occurring in nature.

# The chemical calculus

- A symbolic computation with terms, called molecules.
- A multiset manipulation language: rewriting rules between terms are called reactions.
- Brownian motion justifies the commutative, associative nature of forming pairs of molecules. (Banâtre et al. 2005)

# Example 1

Let $S = \{3, 5, 8, 10, 12\}$. Then

$$(\textit{replace}\ (\langle x \rangle, \langle y \rangle)\ \text{by}\ \langle y \rangle\ \text{if}\ \ x \leq y, \langle 3 \rangle, \langle 5 \rangle, \langle 8 \rangle, \langle 10 \rangle, \langle 12 \rangle)$$

finds the maximum element of the set $S$.

# Example 2

$$
\begin{aligned}
\mathit{largestprime}(6) &= \\
\text{let } \mathit{sieve} &= \mathit{replace}\ (\langle x\rangle, \langle y\rangle) \text{ by } \langle x\rangle \text{ if } x \text{ div } y \text{ in} \\
\text{let } \mathit{max} &= \mathit{replace}\ (\langle x\rangle, \langle y\rangle) \text{ by } \langle y\rangle \text{ if } x \leq y \text{ in} \\
&\quad (\langle \langle 2\rangle, \langle 3\rangle, \ldots, \langle 6\rangle, \mathit{sieve}\rangle, \\
&\quad \mathit{replace}(\langle x\rangle)\ \mathit{by}\ (x, \mathit{max})\ \mathit{if}\ \mathit{true})
\end{aligned}
$$

# Example 2 contd.

$$\langle\langle 2\rangle, \langle 3\rangle, \langle 4\rangle, \langle 5\rangle, \langle 6\rangle, \mathit{replace}\ (\langle x\rangle, \langle y\rangle)\ \text{by}\ \langle x\rangle\ \text{if}\ x\ \text{div}\ y)\rangle \rightarrow$$
$$\langle\langle 2\rangle, \langle 3\rangle, \langle 5\rangle, \langle 6\rangle, \mathit{replace}\ (\langle x\rangle, \langle y\rangle)\ \text{by}\ \langle x\rangle\ \text{if}\ x\ \text{div}\ y\rangle \rightarrow$$
$$\langle\langle 2\rangle, \langle 3\rangle, \langle 5\rangle, \mathit{replace}\ (\langle x\rangle, \langle y\rangle)\ \text{by}\ \langle x\rangle\ \text{if}\ x\ \text{div}\ y\rangle$$

At this point the expression behind the last solution is inert, i.e., contains only solutions and abstractions (*replace*) in the outermost places. Hence, it can be matched with $\langle x\rangle$ in *replace*$(\langle x\rangle)$ *by* $(x, max)$ *if true*.

# Example 2 contd.

Finally,

$$\begin{aligned}(\langle\langle 2\rangle, \langle 3\rangle, \langle 5\rangle, \mathit{sieve}\rangle, \mathit{replace}(\langle x\rangle)\ \mathit{by}\ (x, \mathit{max})\ \mathit{if}\ \mathit{true}) &\rightarrow \\ (\langle 2\rangle, \langle 3\rangle, \langle 5\rangle, \mathit{sieve}, \mathit{max}) &\rightarrow^* \\ (\langle 5\rangle, \mathit{sieve}, \mathit{max}) &\end{aligned}$$

# Basic notions of the chemical calculus 1

More formally,

$$\begin{array}{rcl} M & := & x \\ & | & \gamma(P)[C].M \\ & | & (M_1, M_2) \\ & | & \langle M \rangle \\ P & := & x \\ & | & (P_1, P_2) \\ & | & \langle P \rangle \end{array}$$

$M$ is called a molecule and $P$ is called a pattern, $C$ is a Boolean expression. The term $\langle M \rangle$ is a solution.

# Basic notions of the chemical calculus 2

A redex is a molecule of the form $(\gamma(P)[C].M, N)$. The reduction rule is

$$\gamma(P)[C].M, N \to \phi M, \qquad (\gamma)$$

where $match(P, N) = \phi$ is the unifying substitution between $P$ and $N$ and $\phi(C)$ evaluates to true. A variable unifies with anything, a pair is unified with a pair, and a solution $\langle P \rangle$ is unified with $\langle N \rangle$ provided $N$ is inert, that is, a multiset of $\gamma$-abstractions and solutions.

# Example

$$(\gamma(\langle x\rangle, \langle y\rangle)[(x \leq y)].\langle y\rangle, \langle 3\rangle, \langle 4\rangle, \langle 5\rangle) \rightarrow \langle 4\rangle, \langle 5\rangle,$$

or

$$(\gamma(\langle x\rangle, \langle y\rangle)[(x \leq y)].\langle y\rangle, \langle 3\rangle, \langle 4\rangle, \langle 5\rangle) \rightarrow \langle 3\rangle, \langle 5\rangle$$

depending on which pair in $N$ we choose.

# Basic notions of the chemical calculus 3

Instead of $\gamma$-abstractions, we use the *replace* operator with the rule

$$\textit{replace } P \textit{ with } M \textit{ if } C,$$

where

- $P$ is a pattern which matches the required terms
- $C$ is the reaction condition
- $M$ is the reaction result

*replace* is a shorthand, it can be expressed in the original calculus.

The derived reduction rule for *replace* is

$$\textit{replace } P \textit{ by } M \textit{ if } C, N \rightarrow \textit{replace } P \textit{ by } M \textit{ if } C, \phi(M), \quad (\textit{replace})$$

where the substitution $\phi$ is a matching between $P$ and $N$ and $\phi(C)$ evaluates to true.

# Conclusion

Both models are

- parallel, distributed,
- nondeterministic
- self-organizing systems.

# The Gamma formalism

The Gamma formalism (Banâtre and Métayer, 1993.) is a chemical model of computation.

- The untyped version deals only with multisets as underlying data structures.
- A program is a set of pairs consisting of reactions and actions.
- It is parallel and nondeterministic: designed to be free from artificial sequentiality.

# The Gamma formalism 2

A Gamma-program is an operator transforming multisets into multisets. Let $M$ be a multiset of elements, $R_1, \ldots, R_k$ be conditions, $A_1, \ldots, A_k$ be actions. Then

$$\Gamma((R_1, A_1), \ldots, (R_k, A_k))(M) = \begin{cases} \Gamma(R_1, A_1), \ldots, (R_k, A_k))((M \backslash (x_1, \ldots \\ \qquad \ldots, x_n)) \cup A_i(x_1, \ldots, x_n)), \\ \text{if } x_1, \ldots, x_n \in M \text{ and } R_i(x_1, \ldots, x_n) \\ \text{for some } 1 \leq i \leq k, \\ M \quad \text{otherwise.} \end{cases}$$

**Example**

The function computing the maximal element of a multiset of elements looks like as follows:

$$\begin{aligned} \mathit{maxset}(M) &= \Gamma(R, A)(M) \text{ where} \\ & R(x, y) = (x \leq y) \\ & A(x, y) = (y) \end{aligned}$$

Or

**Example**

$$\begin{aligned} \mathit{primes}(N) &= \Gamma(R, A)(\{2, \ldots, N\}) \text{ where} \\ & R(x, y) = (y \textit{ divides } x) \\ & A(x, y) = (y) \end{aligned}$$

# The Gamma formalism 4

We can grab examples of Gamma-programs from various fields of mathematics: let the predicate *vertices*$(v)$ be true iff $v$ is a set of vertices, moreover, let *singleton*$(w)$ check whether $w$ is a singleton. Then

## Example

$$
\begin{aligned}
\mathit{connected}(G) \quad = \quad & \mathit{singleton}(\Gamma((R_1, A_1), (R_2, A_2))(G)) \text{ where} \\
& R_1(w, v, (m, n)) = \mathit{vertices}(v) \text{ and } \mathit{vertices}(w) \\
& \text{ and } m \in w \text{ and } n \in v \\
& A_1(w, v, (m, n)) = \{v + w\} \\
& R_2(v, (m, n)) = \mathit{vertices}(v) \text{ and} \\
& m \in v \text{ and } n \in v \\
& A_2(v, (m, n)) = \{v\},
\end{aligned}
$$

where $v + w$ is the operation of multiset union. The program *connected* takes at most $|E|$ steps, where $E$ is the set of edges of the graph.

**Example**

The dining philosophers problem:

$$
\begin{aligned}
\mathit{philosopher}(P) \quad = \quad & \Gamma((R_1, A_1), (R_2, A_2))(P) \text{ where} \\
& R_1(f_i, f_j) = (f_j = f_i + 1 \ \mathit{mod}\ n) \\
& A_1(f_i, f_j) = \mathit{phil}_i \\
& R_2(\mathit{phil}_i) = \mathit{true} \\
& A_2(\mathit{phil}_i) = (f_i, f_{i+1}).
\end{aligned}
$$

# Relating membrane systems to chemical programming

- Let $\Pi = (\mathcal{O}, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$ be a P system, possibly with promoter/inhibitor sets for rules and priority relations $(\rho_1, \ldots, \rho_n)$ $(\rho_i \subset R_i \times R_i)$. Assume

$$\mathcal{O} = \{a_1, \ldots, a_k\}$$

and

$$\overline{\mathcal{O}} = \{\overline{a}_1, \ldots, \overline{a}_k\}$$

are co-objects for $\mathcal{O}$.

- A configuration of $\Pi$ is $(\mu, (w_1, \ldots, w_n), (d_1, \ldots, d_n))$, where $w_i : \mathcal{O} \to \mathbb{N}$ and $d_i : \{1, \ldots, n\} \to \{0, 1\}$. If we set $w_i : \mathcal{O} \cup \mathcal{O} \times \{here, in_j, out\} \to \mathbb{N}$, then we get an intermediate configuration.

**Notation**

1. $[x, y] = (\langle x \rangle, y)$
2. $[x_1, \ldots, x_n, x_{n+1}] = [[x_1, \ldots, x_n], x_{n+1}]$

A description is a molecule of the form

$$\begin{aligned} \mathit{Descr} \quad = \quad & [c_{11}, \ldots, c_{1k}, \ldots, c_{m1}, \ldots, c_{mk}, \\ & \overline{c}_{11}, \ldots, \overline{c}_{1k}, \ldots, \overline{c}_{n1}, \ldots, \overline{c}_{nk}, \\ & d_1, \ldots, d_n], \end{aligned}$$

where $c_{ij}$ and $\overline{c}_{ij}$ are natural numbers $(1 \leq i \leq n, 1 \leq j \leq k)$ and $d_i \in \{0, 1\}$ $(1 \leq i \leq n)$.

# Relating membrane systems to chemical programming contd.

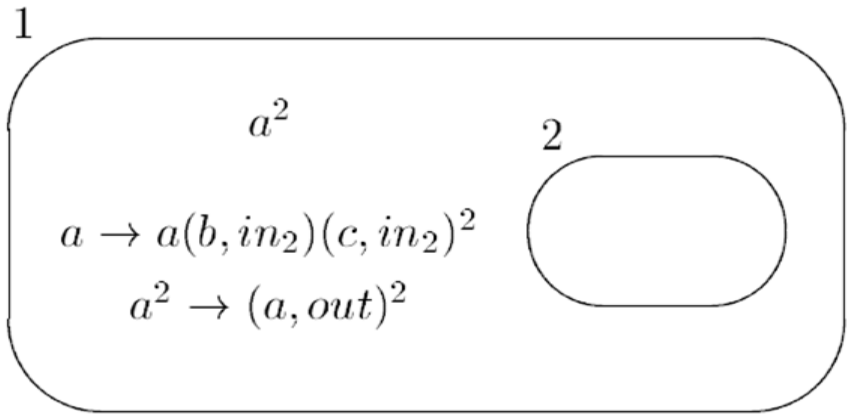
Let $C = (\mu, (w_1, \ldots, w_n), (d_1, \ldots, d_n))$ be an (intermediate) configuration. The description corresponding to $C$ is defined as follows

$$\begin{aligned} Descr((w_1, \ldots, w_n), (d_1, \ldots, d_n)) \quad = \quad & [c_{11}, \ldots, c_{1k}, \ldots, c_{n1}, \ldots, c_{nk}, \\ & \overline{c}_{11}, \ldots, \overline{c}_{1k}, \ldots, \overline{c}_{n1}, \ldots, \overline{c}_{nk}, \\ & d_1, \ldots, d_n], \end{aligned}$$

where $c_{ij} = w_i(a_j)$ and $\overline{c}_{ij} = w_i(a_j, here) + \sum_{p \neq i} w_p(a_j, in_i) + \sum_{\mu(p)=i} w_p(a_j, out)$ with $(1 \leq i, p \leq n)$ and $(1 \leq j \leq k)$. Here $\mu(p)$ denotes the parent membrane of $m_p$. Intuitively, $c_{ij}$ stands for the number of occurrences of $a_j$ in $m_i$, $\overline{c}_{ij}$ denotes the number of $\overline{a}_j$ in $m_i$, while $d_i$ is 1 iff $m_i$ is dissolved or under dissolution.

# Example

An example P system:




$$Descr([[\,]_2]_1, (w_1, w_2), (d_1, d_2)) \quad = \quad [2, 0, 0, 0, 0, 0, 0, 0],$$

where only the value for $c_{11}$, the number of occurrences of $a$ in $m_1$, is non-zero.

# Relating membrane systems to chemical programming contd.

The state corresponding to an (intermediate) configuration is a molecule of the following form

$$\begin{aligned} State((w_1,\ldots,w_n),(d_1,\ldots,d_n),(p_{11},\ldots,p_{nk_n})) &= \\ Descr((w_1,\ldots,w_n),(d_1,\ldots,d_n)) &+ \\ [p_{11},\ldots,p_{1k_1},\ldots,p_{n1},\ldots,p_{nk_n}], & \end{aligned}$$

where $p_{ik_j} \in \{0,1\}$ $(1 \leq i,j \leq n)$ and $+$ denotes the concatenation of two ordered tuples. Intuitively, $p_{ik_j}$ describes the validity of rules: rule $r_{ik_j}$ is valid iff $p_{ik_j} = 1$.

# Relating membrane systems to chemical programming contd.

Steps for the simulation

- Checking rule validity.
- Simulating a maximal parallel step by simulating rule applications one by one.
- Simulating membrane dissolving.
- Removing co-objects and restoring the indicators for rule validity.

# Relating membrane systems to chemical programming contd.

A pattern for a state is a tuple of the form

$$\begin{aligned} S \quad = \quad & [x_{m_1 a_1}, \ldots, x_{m_1 a_k}, \ldots, x_{m_n a_1}, \ldots, x_{m_n a_k}, \\ & \overline{x}_{m_1 a_1}, \ldots, \overline{x}_{m_1 a_k}, \ldots, \overline{x}_{m_n a_1}, \ldots, \overline{x}_{m_n a_k}, \\ & x_{d_1}, \ldots, x_{d_n}, x_{r_{1 k_1}}, \ldots, x_{r_{n k_n}}]. \end{aligned}$$

We simulate the intermediate computational steps in the P system by transitions from one state to another in the chemical calculus. The transition is governed by *replace* operators checking for the actual states by matching them with patterns.

# Relating membrane systems to chemical programming contd.

Firstly, we check rule applicability. Let $r = u \to v \in R_i$. Then we say that $r$ is valid with respect to the configuration $((w_1, \ldots, w_n), (d_1, \ldots, d_n))$ if the following conditions hold:

1. $d_i = 0$
2. $(\forall a \in \mathcal{O})(u(a) \leq w_i(a))$
3. $(\forall a \in prom_r)(w_i(a) \geq 1)$
4. $(\forall a \in inhib_r)(w_i(a) = 0)$
5. $(\forall a \in \mathcal{O})(\forall 1 \leq j \leq n)(v(a, in_j) \geq 1 \supset d_j = 0)$

# Relating membrane systems to chemical programming contd.

Let $r = u \to v \in R_i$, $S$ be a state pattern. Then

$$
\begin{aligned}
Val(r) \quad = \quad & replace\ [S, 0]\ by\ [S[x_r/1], 0]\ if \\
& (x_{d_i} = 0 \wedge \\
& \bigwedge_{1 \le j \le k} (u(a_j) \le x_{m_i, a_j}) \wedge \\
& \bigwedge_{1 \le j \le k} (a_j \in prom_r \supset x_{m_i a_j} \ge 1) \wedge \\
& \bigwedge_{1 \le j \le k} (a_j \in inhib_r \supset x_{m_i a_j} = 0)), \\
& \bigwedge_{1 \le l \le k} \bigwedge_{1 \le j \le n} (v(a_l, in_j) \ge 1 \supset x_{d_j} = 0)),
\end{aligned}
$$

where 0 is a value for synchronization, the role of which to be specified later. We say that rule $r_j \in R_i$ is applicable, if $r_{ij} = 1$.

# Example

In the previous example both rules in $m_1$ are applicable, the pair $[1, 1]$ is appended to the description. From now on, the values for rule validity do not change in the course of the simulation of a maximal parallel step.
If in the example the first rule is applied, then the new configuration is

$$([[\,]_2]_1, ((a, (a, here), (b, in_2), (c, in_2), (c, in_2)), ()), (0, 0)),$$

to which the following description is assigned:

$$[1, 0, 0, 1, 1, 2, 0, 0].$$

# Example contd.

Now we can define an operator which accounts for the transition of states

$$[2,0,0,0,0,0,0,0,1,1] \quad \rightarrow \quad [1,0,0,1,1,2,0,0,1,1]$$

Let $S$ be a state pattern. Then

$$App(r_{11}) \;=\; \textit{replace}\ [S,1]\ \textit{by}$$
$$[S[x_{11}/x_{11}-1, \overline{x}_{11}/\overline{x}_{11}+1, \overline{x}_{22}/\overline{x}_{22}+1, \overline{x}_{23}/\overline{x}_{23}+2], 1]\ \textit{if}$$
$$\big(x_{r_{11}} = 1 \wedge (x_{11} \geq 1)\big).$$

# Relating membrane systems to chemical programming contd.

In general, let $r = u \rightarrow v \in R_i$. Let $S$ be a state pattern. The molecule describing the effect of an execution of $r$ can be defined as follows:

$$App(r) \quad = \quad replace\ [S, 1]\ by\ [\,apply(S, r), 1]\ if \\ \left(x_r = 1 \wedge \bigwedge_{1 \leq j \leq k} (u(a_j) \leq x_{m_i, a_j}\right).$$

# Relating membrane systems to chemical programming contd.

$$apply(S,r)(x_{m_s a_t}) = \begin{cases} x_{m_s a_t} - u(a_t) & \text{if } s = i, \\ x_{m_s a_t} & \text{otherwise}, \end{cases}$$

$$apply(S,r)(\overline{x}_{m_s a_t}) = \begin{cases} \overline{x}_{m_s a_t} + v(a_t, here) & \text{if } s = i, \\ \overline{x}_{m_s a_t} + v(a_t, in_j) & \text{if } s = j \neq i, \\ \overline{x}_{m_s a_t} + v(a_t, out) & \text{if } s = \mu(i), \end{cases}$$

$$apply(S,r)(x_{d_j}) = \begin{cases} 1 & \text{if } v(\delta) = 1, \\ x_{d_j} & \text{otherwise}, \end{cases}$$

$$apply(S,r)(x_r) = x_r \quad \text{if } r \in \mathcal{R}.$$

# Relating membrane systems to chemical programming contd.

When the phase for the simulation of rule application halts, we can turn to the translation of membrane dissolving. Membrane dissolving, $in_j$ and *out* rules leave elements in membranes not existing actually. We move these elements to the parent membranes until an existing membrane is found.

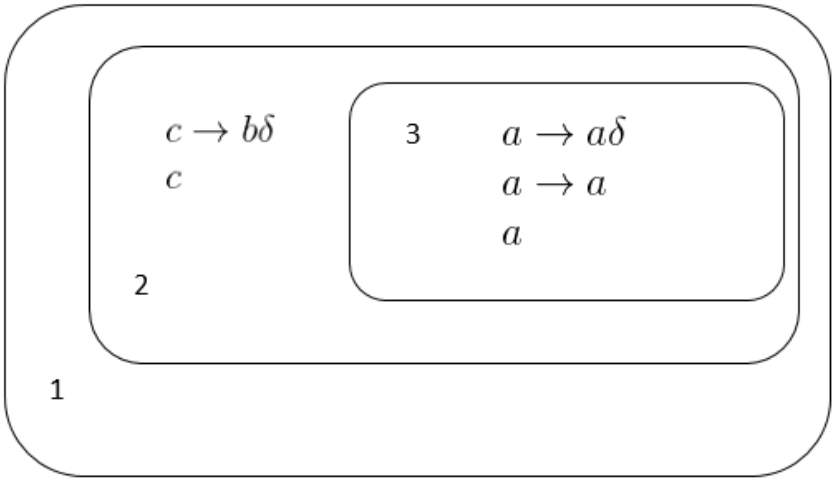
$$\begin{aligned} Dis_i \quad = \quad & \textit{replace } [S,2] \textit{ by } [dis_i(S),2] \textit{ if} \\ & \big(x_{d_i} = 1 \wedge \\ & \big(\bigvee_{1 \le j \le k} x_{m_i a_j} \ge 1 \vee \bigvee_{1 \le j \le k} \overline{x}_{m_i a_j} \ge 1\big)\big), \end{aligned}$$

where

$$dis_i(S)(x_{m_j a_l}) \quad = \quad \begin{cases} x_{m_j a_l} + x_{m_i a_l} & \text{if } j = \mu(i), \\ 0 & \text{if } j = i, \\ x_{m_j a_l} & \text{otherwise.} \end{cases}$$

Similarly for the expressions $dis_i(S)(\overline{x}_{m_j a_l})$.

# Example




If we apply the rules $c \to b\delta$ and $a \to a$ first, then membrane 2 disappears. This is expressed in the following transition of states:

$$[0^3, (0^2, 1), (1, 0^2), 0^3, 0^3, 0^3, 0, 0, 0, 1, 1, 1] \to$$
$$[0^3, 0^3, 0^3, 0^3, (0, 1, 0), (1, 0, 0), 0, 1, 0, 1, 1, 1]$$

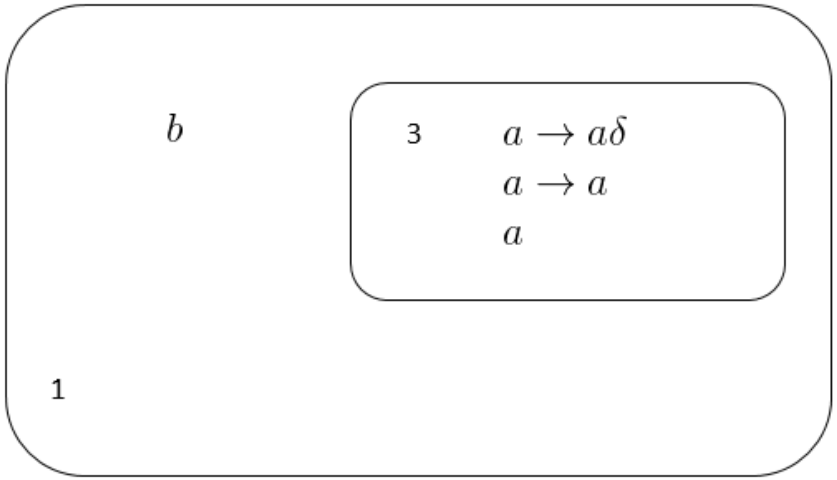
Observe that $x_{d_2} = 1$ in the last state, so we have to move $\overline{b}$ into its parent membrane.

# Example contd.

After restoring the auxiliary tools for simulating a maximal parallel step, we obtain the state

$$[(0,1,0),0^3,(1,0,0),0^3,0^3,0^3,0,1,0,0,0,0]$$

the last value 1 indicating that membrane 2 is missing. This corresponds to the membrane system



# Example contd.

Now rules $a \to a\delta$ and $a \to a$ are applicable, the application of the former followed by the dissolving steps leads to the following reduction sequence:

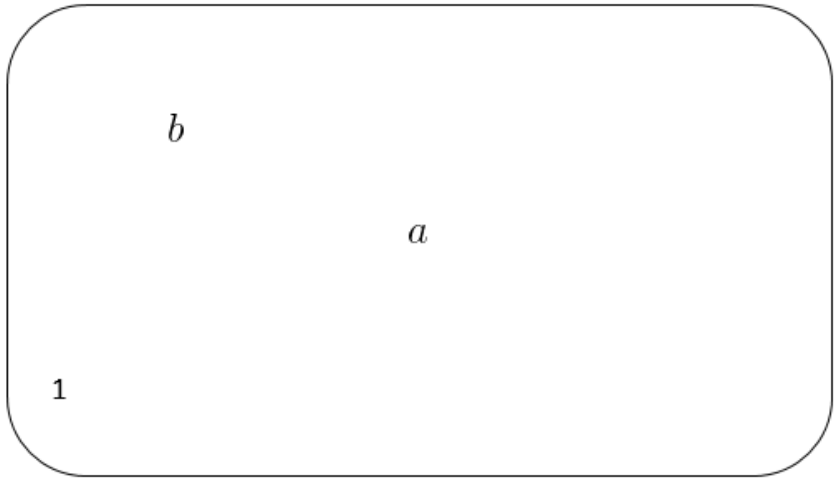
$$
\begin{aligned}
&[(0,1,0),0^3,(1,0,0),0^3,0^3,0^3,0,1,0,0,1,1] \to \\
&[(0,1,0),0^3,0^3,0^3,0^3,(1,0,0),0,1,1,0,1,1] \to \\
&[(0,1,0),0^3,0^3,0^3,(1,0,0),0^3,0,1,1,0,1,1] \to \\
&[(0,1,0),0^3,0^3,(1,0,0),0^3,0^3,0,1,1,0,1,1]
\end{aligned}
$$

# Example contd.

The only task left is to remove the bars from top of the object elements and to restore the indicators of rule applicabilities to their initial values. Executing these steps we obtain

$$[(1, 1, 0), 0^3, 0^3, 0^3, 0^3, 0^3, 0, 1, 1, 0, 0, 0],$$

which yields the membrane system



# Relating membrane systems to chemical programming contd.

We collect the operators necessary for accomplishing the translation:

$$
\begin{aligned}
Val &= \bigcup\{Val(r) \mid r \in \mathcal{R}\},\\
App &= \bigcup\{App(r) \mid r \in \mathcal{R}\},\\
Dis &= \bigcup\{Dis_i \mid i \in \{1,\ldots,n\}\},\\
Rem &= \bigcup\{Rem_{ij} \mid i \in \{1,\ldots,n\}, j \in \{1,\ldots,k\}\},\\
RemVal &= \bigcup\{RemVal(r) \mid r \in \mathcal{R}\},\\
Sync &= replace\ \langle [S, x_{sync}], Val, App, Dis, Rem, RemVal\rangle\ by\\
&\quad \langle [S, x_{sync} + 1\ mod(5)], Val, App, Dis, Rem, RemVal\rangle\ if\\
&\quad \bigvee_{1\le i\le n} x_{r_i} = 1 \vee x_{sync} = 4,\\
&\quad \text{where } S \text{ is a state pattern.}
\end{aligned}
$$

# Relating membrane systems to chemical programming contd.

Let $n_r$ be the number of elements of $\mathcal{R}$. Then

$$M(C_0) \quad = \quad (\langle [State(C_0, 0^{n_r}), 0], Val, App, Dis, Rem, RemVal \rangle, Sync)$$

is a molecule appropriate for the simulation of the computations in the given P system in the following sense.

# The main statement

1. Let $\Pi_0 = (\mathcal{O}, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, (\rho_1, \ldots, \rho_n))$ be a P system of order $n$ with membrane dissolving, promoter/inhibitor sets for rules and priority relations. Assume

$$C_0 = (w_1, \ldots, w_n) \Rightarrow^* C_1 = (w'_{n_1}, \ldots, w'_{n_i}),$$

where $1 \leq n_1 \leq \ldots \leq n_i \leq n$. Let

$$w'_j = \begin{cases} w_{n_l} & \text{if } j = n_l \text{ for some } 1 \leq l \leq i, \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, let $d'_j = 1$ iff $j \notin \{n_1, \ldots, n_i\}$. Set $C'_1 = ((w'_1, \ldots, w'_n), (d'_1, \ldots, d'_n))$. Then

$$M(C_0) \rightarrow^* M(C'_1).$$

If the computation starting from $\Pi_0$ contains at least one step, then reduction sequence starting from $M(C_0)$ is non-empty either.

2. Let $\Pi_0$ and $M(C_0)$ be defined as above. Assume

$$M(C_0) \rightarrow^* M_1$$

such that $M_1 = M((w'_1, \ldots, w'_n), (d'_1, \ldots, d'_n))$, where $d'_i = 1$ implies $w'_i = 0$ and $p_{jt} = 0$ $1 \leq t \leq k_j$ and $1 \leq j \leq n$. Let $1 \leq n_1 \leq \ldots \leq n_i \leq n$ be the indices of $d'_j$ with $d'_j = 0$. Then there exists a P system $\Pi_1$ with membranes labelled $m_{n_1}, \ldots, m_{n_i}$ and configuration $(w'_{n_1}, \ldots, w'_{n_i})$ such that

$$C_0 = (w_1, \ldots, w_n) \Rightarrow^* C_1 = (w'_{n_1}, \ldots, w'_{n_i}).$$

Moreover, if the length of $M(C_0) \rightarrow^* M_1$ is at least one, then the length of the computation starting from $\Pi_0$ is non-zero.

# Relating membrane systems to chemical programming contd.

**Corollary**

*Let* $\Pi = (\mathcal{O}, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, (\rho_1, \ldots, \rho_n))$. *Then* $\Pi$ *is strongly (resp. weakly) normalizing iff* $M(w_1, \ldots, w_n)$ *is strongly (resp. weakly) normalizing.*

# Open questions, future work

- The reverse translation (from the Gamma formalism into the membrane systems) is still under construction.
- There are many formalism of chemical computation in the literature. How they are related to each other?
- How chemical computation is related to the original $\lambda$-calculus? What kind of reduction strategies are obtained by defining different transformations?

# Bibliography


- J.P. Banâtre, P. Fradet, Y. Radenac, Principles of chemical computing. *Electronic Notes in Theoretical Computer Science* 124 (2005) 133–147.
- J.P. Banâtre, D. Le Métayer, Programming by multiset transformation. *Communications of the ACM* 36 (1993), 98–111.

# Acknowledgements


- The Hungarian Scientific Research Fund "OTKA" grant no. K75952
- The TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project, co-financed by the European Social Fund