P Systems with Reactive Membranes

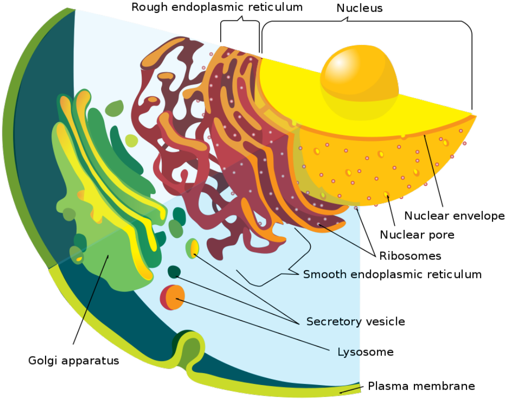
Artiom Alhazov Rudolf Freund Sergiu Ivanov
David Orellana-Martín Antonio Ramírez-de-Arellano
José Antonio Rodríguez Gallego

The Extended Moldovan Team
The Sevillian Team

CMC 2023

P systems vs. Life

Inspired by the eukaryotic cell | Decentralized computing



Membrane 1:

$a \to aa$

$a \to (a, \text{out})$

a

Membrane 0:

$a \to b$

$b \to c\,(c, \text{in})$

Use P systems as a tool for thinking about Life.

Emergence of Life

↪ Emergence of multiple elements:

- organic compounds
- catalytic cycles
- milieu separations $\Longrightarrow$ membranes!
- genetic code

Roadmap:

1. Capture the emergence of membranes.
2. Go further.

Problem

First-class membranes

Membranes are *already* in the definition.

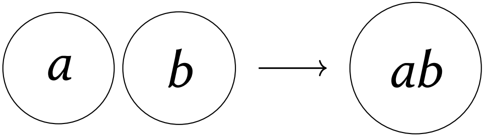
How to do emergence of membranes?

Very simple membranes $\rightsquigarrow$ More complex membranes

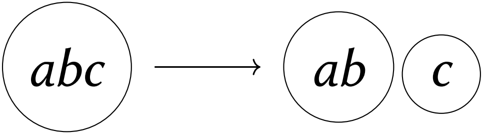
Emergence of space

1. Symbols carry space.
2. Regions emerge from symbol interaction.

Merge:



Split:



Evolution rules

$$u \to v \qquad u, v \in V^*$$

Uniformity: Same rules in all membranes.

Locality: All symbols in u must belong to the same membrane.

Evolution rules
|
on top of
|
Splitting & merging

Chemistry
|
on top of
|
Space

Reactive membranes

Formal definitions

P systems with reactive membranes

$$\Pi = (O, T, W_0, R, \delta)$$

- O: the alphabet of objects
- $T \subseteq O$: the alphabet of terminal objects
- $W_0 \subseteq \mathcal{P}_{fin}(O^\circ)$: the initial finite set of multisets
- $R \subseteq O^\circ \times O^\circ$: the set of evolution rules
- δ: the derivation mode

$$\forall u \to v \in R : \; u \neq \lambda \;\vee\; v \neq \lambda$$

Salient features

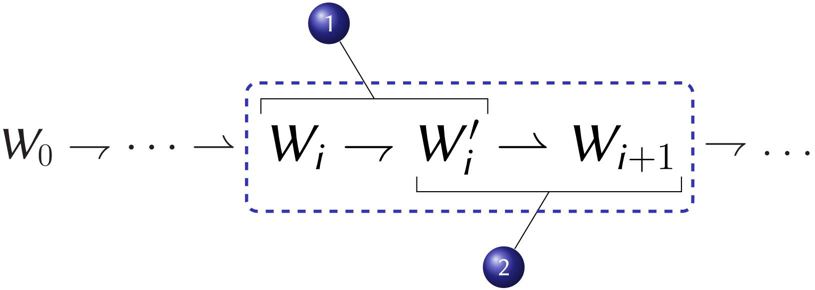
1. No explicit membrane structure: $W_0 \subseteq \mathcal{P}_{fin}(O^\circ)$
   - membrane $\sim$ individual multisets
   - no membrane nesting
2. One set of rules for all membranes.

Computation

Configuration: $W_i \subseteq \mathcal{P}_{fin}(O^\circ)$

Computation step:

1. Splitting & merging stage $\rightharpoondown$
2. Evolution stage $\rightharpoonup$



Splitting & merging $W_i \rightharpoonup W_i'$

1

Non-deterministically partition W_i:

$$W_i = M_i \cup S_i \cup I_i$$

- M_i: the multisets to merge
- S_i: the multisets to split
- I_i: the multisets to keep intact

- $|M_i|$ is even
- $S_i \cap M_i = S_i \cap I_i = M_i \cap I_i = \emptyset$

Splitting & merging $W_i \rightharpoonup W_i'$

2

Partition M_i into a set of disjoint pairs, and merge each of the pairs.

1. Non-deterministically pick a bijection $\varphi : [1..|M_i|] \to M_i$.
2. Set $\hat{M}_i = \{(\varphi(2k-1), \varphi(2k)) \mid 1 \leq k \leq |M_i|/2\}$.
3. Set $M_i' = \{w_1 \cup w_2 \mid (w_1, w_2) \in \hat{M}_i\}$.

Splitting & merging $W_i \rightharpoonup W_i'$ 3

1. The set of all possible ways to split a multiset: $\mathrm{split}(w) = \{(w_1, w_2) \mid w_1 \cup w_2 = w, w_1, w_2 \in O^\circ\}$.
2. The set of all possible ways to split the multisets in S_i:
$$\hat{S}_i = \prod_{w \in S_i} \mathrm{split}(w).$$
3. Non-deterministically pick $S_i' \in \hat{S}_i$.

Splitting & merging $W_i \rightharpoonup W_i'$

4

Collect the results of splitting and merging:

$$W_i' \;=\; M_i' \,\cup\, \mathrm{flatten}(S_i') \,\cup\, I_i$$

- $\mathrm{flatten}(S_i') = \{w_1, w_2 \mid (w_1, w_2) \in S_i'\}$

Evolution $W_i' \rightharpoonup W_{i+1}$

$$W_{i+1} = \{\, w \mid \underbrace{w' \xRightarrow{\delta, R} w}, \; w' \in W_i' \,\}$$

derive a multiset w' from w by applying the rules from R according to the mode δ

Halting

W_i is halting if no more rules are applicable after any splitting & merging:

$$\forall W_i' : W_i \rightharpoondown W_i' \quad \forall w' \in W_i' : w' \not\xrightarrow{\delta,R}$$

A halting computation ends in a halting configuration.

Result of a computation

Restrict everything in a halting configuration W_n to the terminal alphabet T:

$$\left(\bigcup_{w \in W_n} w \right) \Bigg|_T = \bigcup_{w \in W_n} w|_T$$

$$w|_B(a) = \begin{cases} w(a) & \text{if } a \in B \\ 0 & \text{otherwise} \end{cases}$$

Examples

Example 1

max

$r_1 : ab \to d$ $r_2 : abc \to f$ $r_3 : a \to e$

a b c $T = \{d, f\}$

1. $\{a, b, c\} \rightharpoondown \{a, b, c\} \xrightarrow{r_3} \{e, b, c\}$ Result: Λ (empty multiset)

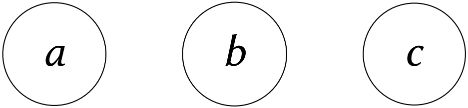
   No other rules ever applicable.

2. $\{a, b, c\} \rightharpoondown \{ab, c\}$
   - $\xrightarrow{r_1} \{d, c\}$ Result: d
   - $\xrightarrow{r_3} \{eb, c\}$ Result: Λ

Example 1

max

$r_1 : ab \rightarrow d$ $\qquad$ $r_2 : abc \rightarrow f$ $\qquad$ $r_3 : a \rightarrow e$



$T = \{d, f\}$

3. r_2 never applicable with $W_0 = \{a, b, c\}$
   - Two mergers are required, but a is necessarily consumed before by r_3 or r_1.

4. $\{ab, c\}$, $\{abc\}$ → $\{abc\}$

   $\{abc\} \xrightarrow{r_1} \{dc\}$ Result: d

   $\{abc\} \xrightarrow{r_2} \{f\}$ Result: f

   $\{abc\} \xrightarrow{r_3} \{ebc\}$ Result: Λ

Notations

Conclusion: The number of initial parts $|W_0|$ matters.
non-empty

$Re_n OP(\delta, \tau)$: The P systems with reactive membranes with $|W_0| = n$, running under the mode δ, and using rules of type $\tau \in \{coo, ncoo\}$.

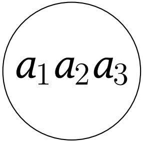
$NRe_n OP(\delta, \tau)$: The number languages generated by $Re_n OP(\delta, \tau)$.

$PsRe_n OP(\delta, \tau)$: The multiset languages generated by $Re_n OP(\delta, \tau)$.

Example 2

max

$$r_{1,2,3} : a_i \to a_i' \quad r_{4,5,6} : a_i' \to a_i'' \quad r_7 : a_1''a_2''a_3'' \to f$$



$$T = \{a_1'', a_2'', a_3'', f\}$$

1. $$\{a_1 a_2 a_3\} \longrightarrow \{a_1 a_2 a_3\} \xrightarrow{r_{1,2,3}} \{a_1' a_2' a_3'\} \longrightarrow$$
$$\longrightarrow \{a_1' a_2' a_3'\} \xrightarrow{r_{4,5,6}} \{a_1'' a_2'' a_3''\} \longrightarrow \{a_1'' a_2'' a_3''\} \xrightarrow{r_7} \{f\}$$

2. $$\{a_1 a_2 a_3\} \longrightarrow \{a_1, a_2 a_3\} \xrightarrow{r_{1,2,3}} \{a_1', a_2' a_3'\} \longrightarrow$$
$$\longrightarrow \{a_1', a_2', a_3'\} \xrightarrow{r_{4,5,6}} \{a_1'', a_2'', a_3''\}$$

Splitting & merging may prevent the applicability of a rule with $|\mathrm{LHS}| \geq 3$.

Computational power

W_0 can be extended by any number of additional Λ-vesicles:

Lemma

For every $\Pi \in Re_1OP(\delta, \tau)$ there exists an equivalent $\Pi' \in Re_nOP(\delta, \tau)$ system, for every $n > 1$.

Proof sketch

From $\Pi = (O, T, \{w\}, R, \delta)$ construct
$\Pi' = (O, T, \{w, w_2 = \Lambda, \ldots, w_n = \Lambda\}, R, \delta)$.

1. No additional rule applications in $w_2, \ldots, w_n$.
2. $\forall i \in [2..n] : w \cup w_i = w$.

Halting and *ncoo*

Remark

Configuration W is halting $\iff$
no more rules are applicable to flatten(W).

All rules are non-cooperative, i.e.,
only one symbol on the left-hand side of rules $\Rightarrow$
splitting and merging need not be considered.

Splitting & merging and *ncoo*

Theorem 1

For any $\delta_1, \delta_2 \in \{asyn, seq, max, smax\}$, $Y \in \{N, Ps\}$, and any $n \geq 1$:

$$YRe_nOP(\delta_1, ncoo) = YOP_1(\delta_2, ncoo) = Y\mathcal{L}(REG).$$

Proof sketch

1

$YOP_1(\delta_2, ncoo) = Y\mathcal{L}(REG)$ is folklore  .

We argue that, for any $\delta_1 \in \{asyn, seq, max, smax\}$,
$YRe_nOP(\delta_1, ncoo) = YOP_1(asyn, ncoo) = Y\mathcal{L}(REG)$.

Splitting & merging and *ncoo*

Theorem 1

$$YRe_nOP(\delta_1, ncoo) = YOP_1(\delta_2, ncoo) = Y\mathcal{L}(REG).$$

Proof sketch

2

Prove $YRe_nOP(\delta_1, ncoo) = YOP_1(asyn, ncoo)$

($\Rightarrow$) For $\Pi' = (O, T, \{\mathrm{w}_1, \ldots, \mathrm{w}_\mathrm{n}\}, R, \delta_1)$ construct

$\Pi = (O, T, w_1 \cup \ldots \cup \mathrm{w}_\mathrm{n}, R, asyn) \in OP_1(asyn, ncoo)$.

- For $\delta_1 = seq$, Π' may feature some kind of *smax*, but it does not matter because of *ncoo*.

Splitting & merging and *ncoo*

Theorem 1

$$YRe_nOP(\delta_1, ncoo) = YOP_1(\delta_2, ncoo) = Y\mathcal{L}(REG).$$

Proof sketch

3

Prove $YRe_nOP(\delta_1, ncoo) = YOP_1(asyn, ncoo)$.

$(\Leftarrow)$ For $\Pi = (O, T, w, R, asyn)$ construct
$\Pi' = (O, T, \{w\}, R, asyn) \in Re_1OP(asyn, ncoo)$.

- Π' simulates Π by never splitting.
- Π' cannot apply more rules than Π.

Partially Blind Register Machines PBRM

Registers machines with two types of instructions:

$(p, \mathrm{ADD}(r), q, s)$: in state p increment register r and jump to state q or state s.

$(p, \mathrm{SUB}(r), q)$: in state p try to decrement register r; if successful, jump to state q, otherwise abort the computation without producing a result.

PsPBRM: The multiset languages generated by PBRMs.

$PsBRM \subseteq$ Reactive membranes + coo

Theorem 2

$PsPBRM \subseteq PsRe_1OP(\delta, coo), \delta \in \{asyn, seq, max, smax\}$.

Proof idea

Simulate $(p, \text{ADD}(r), q, s)$ by $p \to qa_r$ and $p \to sa_r$.

Simulate $(p, \text{SUB}(r), q)$ by $pa_r \to q$, $p \to p$, $a_r \to a_r$.

- [1] p and a_r are in the same membrane $\Rightarrow$ decrement.
- [2] No more a_r anywhere, unit rule p → p is applied $\Rightarrow$ no halting.
- [3] p and a_r in different membranes: unit rules are applied in branches in which they do not meet, but $\exists$ a branch in which p and a_r did not get separated in the first place; in this branch the decrement happens.

Extensions

Limiting the membrane size

Forbid membranes containing more than K symbols.

Possible semantics:

1. Prohibit rule applications adding more symbols.
2. Force the membrane to split.

Probably no impact if $K > \max_{LHS}|LHS|$.

Rules travel like objects

1. Make $\mathcal{C} \subseteq \mathcal{P}_{fin}\left(\left(O \cup R\right)^\circ\right)$.
2. Objects and rules are split and merged.
3. In the evolution substep, evolve the atomic symbols from each $w \in \mathcal{C}$ by the rules present in w, according to the mode δ.

Splitting and merging of rules

A rule $u \rightarrow v$ can split into $u \rightarrow \alpha$ and $\alpha \rightarrow v$.

Two rules $u \rightarrow \alpha$ and $\alpha \rightarrow v$ can merge into $u \rightarrow v$.

- $u, v, \alpha \in O^\circ$

Origins of Life?

Discussion

Splitting & merging

Easy to imagine, difficult to define and work with.

Modulate computational power in interesting ways.

No modelling

P systems with reactive membranes are not a model as understood in biological modelling.

P systems with reactive membranes are a formal vehicle for thinking about the origins of Life.

Relationship to other P system variants

- active membranes
- mobile membranes
- vesicles of multisets

Difference: compulsory splitting and merging.
↪ emergence of a basic form of space.

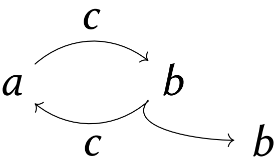
Topology		Geometry
|	vs.	|
Space = neighborhoods		Space = coordinates

Open questions

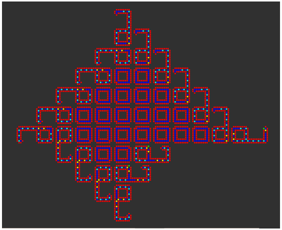
Back to the origins

Next steps in thinking about the origins of Life:

1. Implement catalytic cycles.



2. Implement self-replication.



More on computational power

Splitting & merging has no effect on *ncoo*.

Can splitting & merging augment the computational power? in which cases?

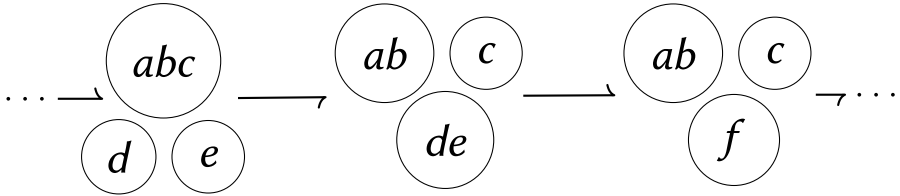
Halting vs. retrieving the result

Halting relies on binary mergers.

Retrieving the result flattens (merges) everything.

Asymmetry

The computational power depends on the definition of halting and the procedure for retrieving the result.



$\forall \delta_1, \delta_2, \delta \in \{asyn, seq, max, smax\}, \forall Y \in \{N, Ps\}, \forall n \geq 1$:

Th 1: $YRe_nOP(\delta_1, ncoo) = YOP_1(\delta_2, ncoo) = Y\mathcal{L}(REG)$.

Th 2: $PsPBRM \subseteq PsRe_1OP(\delta, coo)$.

Not a model

A formal vehicle

Topology | Space = neighborhoods

Geometry | Space = coordinates

Computing power? Catalytic cycles? Self-replication?

Thank you Chema! Thank you BWMC organizers!