Membrane and metabolic systems

Dr Giuditta Franco

Department of Computer Science, University of Verona, Italy

Formal Languages

Over an *alphabet* A, we have a the *linguistic universe* $A^\star = \cup_{n\in\mathbb{N}} A^n$ where A^n is defined by induction: : $A^0 = \{\lambda\}$ and $A^{n+1} = A \cdot A^n$.

A *formal language* is an element of $\mathcal{P}(\mathcal{A}^\star)$, that is, a collection of strings. Examples of languages: bipartite, bisomatic, trisomatic, duplicates, prefix, suffix, fix length codes.

Operations on languages: sum, intersection, difference, complement, concatenation, power, Kleene star.

Language as a discrete function, language membership as a solution to combinatorial problems.

Two main ways to define a language

- **Logic-algebraic**: set-based or by patterns, such as $3x + 4y$, with x and y variables on $A^\star$, or $\{a\}^\star\{b + c\}c^4$ (monosomatic a^n, bisomatic $a^n b^n$, trisomatic $a^n b^n c^n$)
- **Algorithmic**: by *grammars* (language generators) or by automata (language recognizers).

Chomsky grammar

$$G = (A, T, S, R)$$

A alphabet, $T \subset A$ terminal [1] alphabet ($N = A \setminus T$), $S \in N$ starting symbol, R binary relation on $A^\star$.
Namely, if $(\alpha, \beta) \in R$, then $\alpha \notin T^\star$, and we denote it $\alpha \to \beta$.

One-step rewriting: $\phi \Rightarrow_G \psi$, if $\phi = x\alpha y$, $\psi = x\beta y$, $(\alpha, \beta) \in R$
Multiple-steps rewriting [2] (transitive closure): $\phi \Rightarrow^\star_G \phi$, if $\exists \phi_1, \phi_2, \ldots, \phi_n$ such that $\phi = \phi_1$, $\psi = \phi_n$, and $\phi_i \Rightarrow_G \phi_{i+1} \forall i$

$$L(G) = \{\beta \mid \beta \in T^\star, S \Rightarrow^\star_G \beta\}$$

[1] halt criterion

[2] Conventiona rewriting systems are non-deterministically sequencial. Unconventional ways to apply these rules: probabilistic, by priority relation, (maximal, conditionated) parallel.

Examples

$\{S \to a, S \to aS\}$ generates monosomatic language

$\{S \to ab, S \to aSb\}$ generates bisomatic language

$\{S \to aS, S \to Sb, S \to \lambda\}$ generates bipartite language

$\{S \to abc, S \to aSBc, cB \to Bc, bB \to bb\}$ generates trisomatic language

$\{S \to abc, S \to aaBbc, Bb \to bbc, Cb \to bC, Cc \to cc, Cc \to Dcc, bD \to Db, aD \to aaB\}$ .. homework

Post systems

Post rewriting system to generate the trisomatic language, with variables $x, y \in \{a, b, c\}^{\star}$:

$$\frac{xaby}{xaabbyc}$$

with axioms $x = \lambda$ and $y = c$. (second round x=a and y =bcc)

$$\frac{axbyc}{aaxbbycc}$$

with axioms $x = y = \lambda$. (second round x=ab and y =c)

Four types of grammars/languages

Type 0: $\alpha \in A^\star NA^\star$ (general form)

Type 1: (type 0 AND) $|\alpha| \leq |\beta|$ (monotonic form)

Type 2: (type 1 AND) $|\alpha| = 1$ (context-free form)

Type 3: (type 2 AND) $\beta \in T \cup TN$ (regular form).

For i=0,1,2,3, a grammar is of type i if all its rules are of type i, $\mathcal{L}_i$ is the class of languages generated by grammars of type i.

By definition, $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$, and we say a language L is of type i if $L \in \mathcal{L}_i \backslash \mathcal{L}_{i-1}$

Four types of FLT results

Equivalence: $DFSA = FSA$

Decidability [3]: languages in $\mathcal{L}_1$ are decidable.

Closure: $\mathcal{L}_3$ is closed by $\star$, by union, by complement, by intersection.

Normalization: $REG = \mathcal{L}_3$, where REG is defined as $Close(A, ;, +, \star)$.

[3] Intuitive concepts of computable, decidable, tractable - difference between computability and complexity.

Chomsky hierarchy

$$\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$$

By definition, $CF = \mathcal{L}_2$ and $CS = \mathcal{L}_1$.

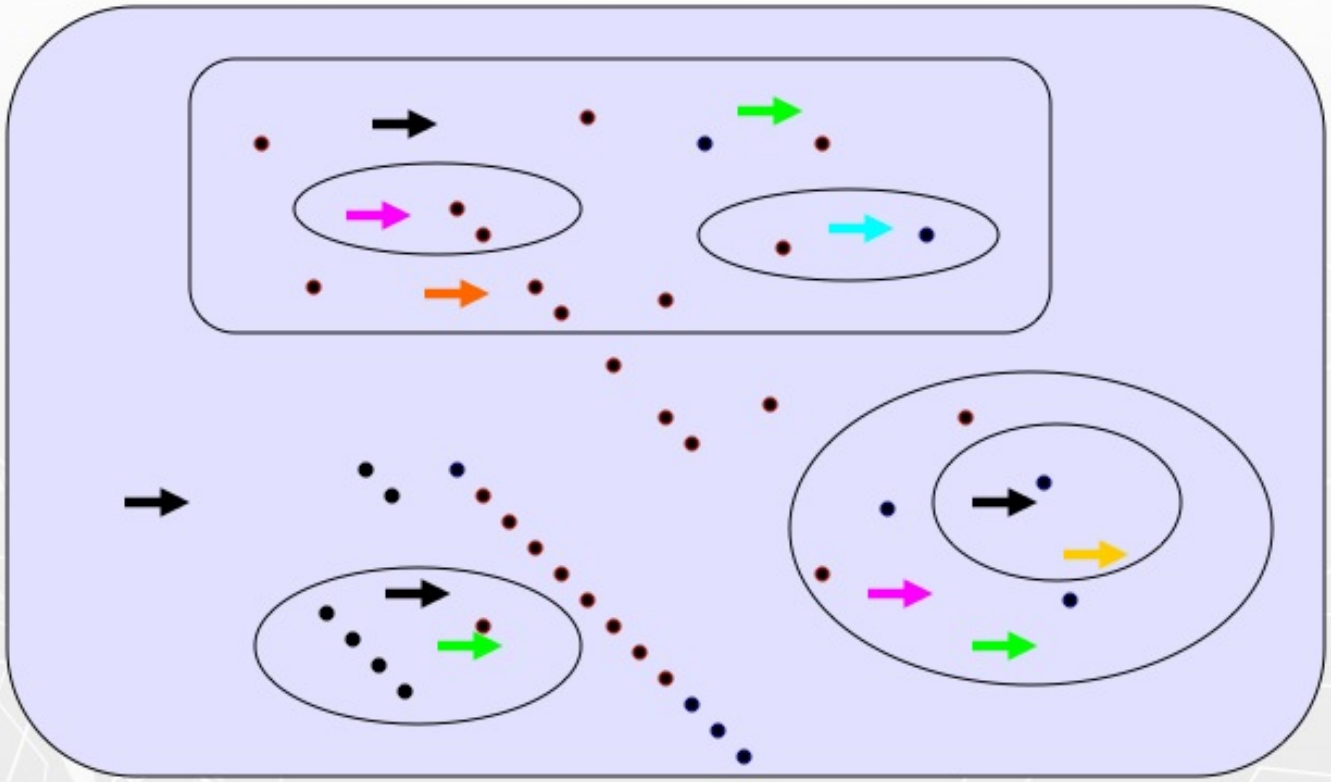
Central theorem of representation: $RE = \mathcal{L}_0$. For any semidecidable (Recursively Enumerable) language there exists a (general) grammar generating it.

We will prove that:

$$REG \subset CF \subset CS \subset RE$$

Remark: $|\mathcal{P}(A^\star)| > |\mathcal{L}_0|$

Paun 1998 **From FLT Tradition**



[[-->||...||-->]] *implies* Computational Universality

Courtesy of prof. Manca, Univ. VR, IT

Membrane computing

Inspired by the architecture, the functioning, the properties of the living cell and their constituents, P systems (membrane systems) are defined as a **hierarchical arrangement of regions where multisets of objects evolve according to different evolutionary rules.**[4]

Aims: powerful and effective tools for computation, to better understand the nature of computation through simulation of biochemical/cell systems behaviour.

[4] The Oxford Handbook of Membrane Computing, 2010.

Basic (symbol-object) definition of P systems ['98]

A transition P system of degree m, with $m \geq 1$, is a construct

$$\Pi = (\Sigma, T, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m)$$

where:

- Σ (or A, or O) is the alphabet (its elements are called objects)
- $T \subseteq \Sigma$ (output alphabet, of terminal objects)
- $\mu \in \mathbb{N} \times \mathbb{N}$ is the membrane structure, with m labeled membranes
- $\{w_i\}_{i=1,\ldots,m}$ are multisets associated to the ith-regions of μ
- $\{R_i\}_{i=1,\ldots,m}$ are finite sets of evolution rules (rewriting rules over multisets on Σ) associated to the ith-regions of μ.

Essential features

Space/matter duality, typical of discrete systems, here it is membrane/objects.

Multiset rewriting rules are located in regions, and they act on objects present in the region where the rules reside: rewriting rules on commutative strings are applied by maximal parallelism (prob., priority, determinism).

Main new ingredients: *multiset* rewriting, *parallel* strategy, *membrane* structure.

Multiset rewriting rules

- $u \to v$, where u and v are strings (representing multisets of objects from a given set O)
- e.g. $a^2bc^3 \to bd^4$ transforms two copies of a, one of b, and three of c into four copies of d, while b is reproduced (it plays here the role of a catalyst),
- this rule can be applied to the multiset $a^3bc^6d^2$, but not to abc^6d^2, because $a^2bc^3 \subseteq abc^6d^2$ does not hold

Multiset rewriting rules (contd.)

- For a rule $u \to v$ we say that $|u|$ (the weight of the multiset on the left hand side) is the *weight* of the rule; a rule with the weight at least two is said to be *cooperative*.
- A rule of weight 1 is called *non-cooperative*.
- *Catalytic rules* are cooperative rules of the form $ca \to cv$, where c and a are objects with $c \neq a$, v a multiset, and c does not appear in v (c only assists a in evolving into v, but c itself is not undergoing any transformation).

Maximal parallelism

- rules should be used in parallel to the maximum degree possible
- assume that currently the multiset of objects present in region h is $w = a^3b^2c^2$ and the set of rules R residing in h is $R = \{r_1, \ldots, r_5\}$, with $r_1 : ab \to v_1$, $r_2 : c \to v_2$, $r_3 : bc \to v_3$, $r_4 : a^3c^2 \to v_4$, $r_5 : ad \to v_5$, for some $v_1, \ldots, v_5$.
- Applying r_1 removes (consumes) the multiset ab, while two parallel applications of r_1 remove the multiset a^2b^2. The remaining still available multiset ac^2 does not allow one more parallel application of r_1, but it allows for a parallel application of r_2, it even allows two parallel applications of r_2, which remove the multiset c^2. The remaining multiset a does not allow an application of any rule at all, and therefore we say that the parallel application of the multiset of rules $\{(r_1, 2), (r_2, 2)\}$ is maximally parallel.

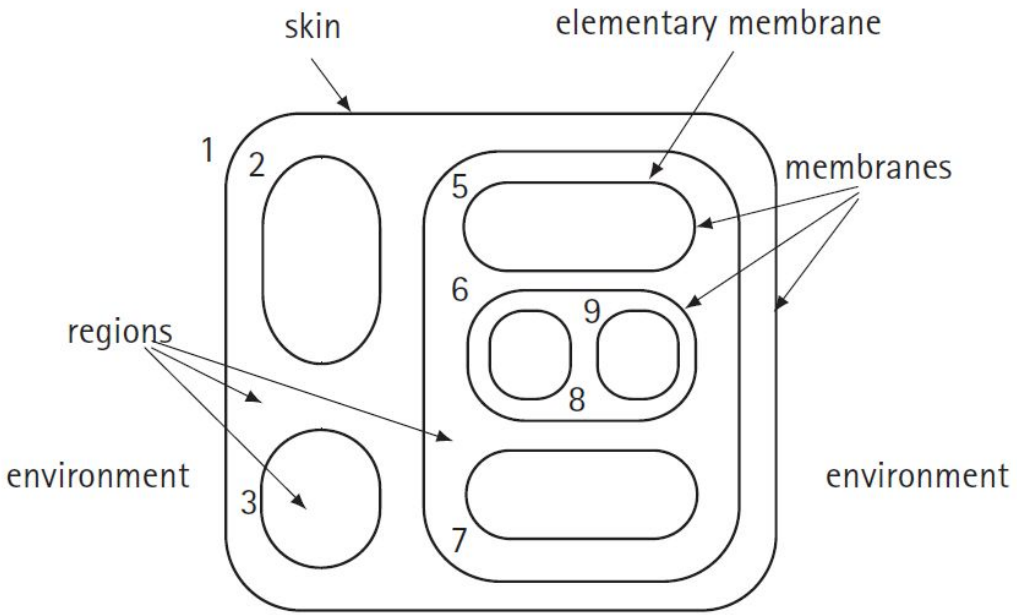
Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Maximal parallelism and non-determinism

- Parallelism: More than one rule may be applied (on different objects) in the same step
- Multiset: each rule may be applied more than once in the same step (on different objects)
- Maximality: A multiset is chosen non-deterministically such that no other rule can be applied to the system obtained by removing all the objects involved by that choice.

Example: Given $\begin{cases} c \to a \\ a \to b \\ a \to bc \\ b \to bc \\ b \to ab \end{cases}$ of rules r_1, r_2, r_3, r_4, r_5,
and starting from multiset *abccab*: how many multisets you obtain by applycation of the above rules by non-det. max. par?

Membrane structures (contd.)



Notation $[_1\,[_2\,]_2[_3\,]_3[_4\,[_5\,]_5[_6\,[_8\,]_8[_9\,]_9\,]_6[_7\,]_7\,]_4\,]_1$

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

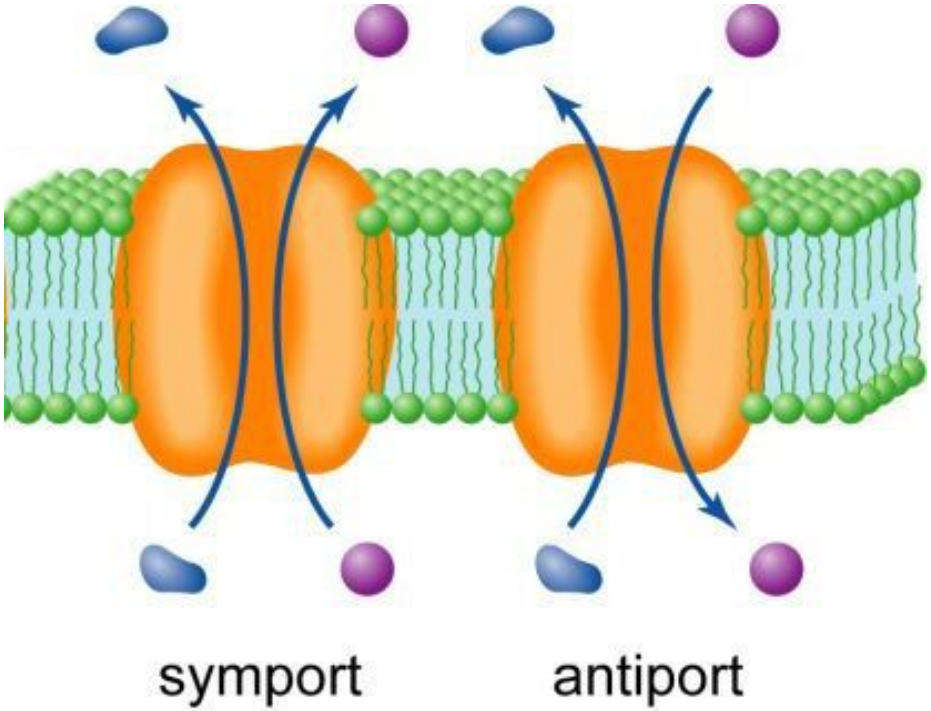
Role of the membrane

Each membrane is addressable by an attached label – regions are in one-to-one correspondance with membranes

Membranes as separation, selection, concentration protection: selectively permeable barriers between the cell interior and the environment around the cell, they govern the bidirectional movement of chemicals between the cell and the outside.

Therefore, membranes are different than registers, as they have an active role in computation.

Symport/antiport P systems (contd.)



Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Multiset rewriting rules (contd.)

- The communication between regions is provided by passing/exchanging objects between regions through membranes that separate them. Such a communication is a basic mechanism of cooperation between regions, and it is incorporated in rewriting rules by *target indications*.
- The rules are of the form $u \rightarrow v$, where u is a string over O, and v is a string over $O \times \{here, out, in\}$. Thus, each element of v is of the form (a, tar), where a is an object from O and tar is either $here$, out or in.
  - If $tar = here$, then a remains in the region where the rule resides.
  - If $tar = out$, then a is sent to the upper neighbour of the region where the rule resides (recall that the upper neighbour of the skin region is the environment).
  - If $tar = in$, then a is sent non-deterministically to one of the lower neighbours.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Multiset rewriting rules (contd.)

- if h is not elementary, $a^2bc^3 \rightarrow ba^2c(da, out)(ca, in)$ means that the rule is applicable if h contains at least two objects a, one object b, and three objects c
- the application of this rule
  - removes from h two objects a, one object b, and three objects c
  - produces one object b, four objects a, two objects c, and one object d
  - the produced object b remains in h, two produced objects a remain in h, one produced object c remains in h, the produced object d and one produced object a are sent to the upper neighbour, one produced object a is sent to a (non-deterministically chosen) lower neighbour, and one produced object c is sent to a (non-deterministically chosen) lower neighbour

Multiset rewriting rules

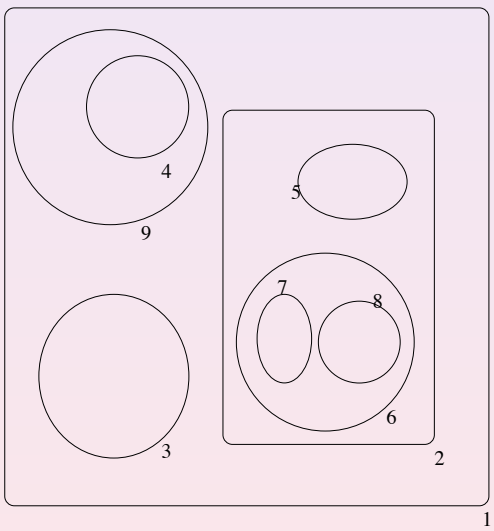
If *h* is the skin region, then the produced object *d* and one produced object *a* are sent to the environment

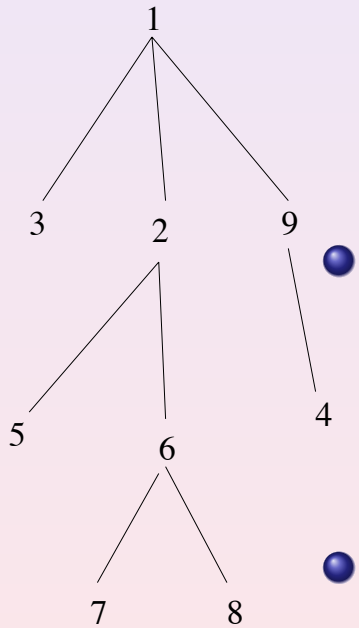
Since there are no evolution rules placed in the environment, the skin region never receives objects from the environment and therefore objects that leave the membrane structure are lost (can never return)

In some cases, environment is supposed to *feed* the system by providing a constant amount of certain elements.

Rules changing the membrane structure

A *b*asic P system with active membranes [F. Bernardini et al. 2004] is $\Pi = (V, K \cup \{1\}, \mu, w_1, \ldots, w_m, R)$





- Transformation, out, in, rules $[_i a \to v]_i$, $[_i a]_i \to b[_i]_i$, $a[_i]_i \to [_i b]_i$, $a, b \in V$, $v \in V^\star$, $i \in K \cup \{1\}$;
- Membrane dissolving rules $[_i a]_i \to a$ (notation: $a \to a\delta$) with $a \in V$, and $i \in K$;
- Creation (division, duplication) rule $[_i a \to [_j b]_j]_i$ with $a, b \in V$, and $i \in K \cup \{1\}$ and $j \in K$.

Symbol-object P systems – The basic variant (contd.)

Definition 1

$\Pi = (O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o)$, where

1. O is an alphabet (its elements are called *objects*);
2. μ is a membrane structure consisting of m membranes, with the membranes (and hence the regions) injectively labelled by $1, 2, \ldots, m$, m is called the *degree* of Π;
3. ω_i, $1 \leq i \leq m$, are strings that represent multisets over O associated with regions $1, 2, \ldots, m$ of μ;
4. R_i, $1 \leq i \leq m$, are finite sets of *evolution rules* over O; R_i is associated with region i of of μ; an *evolution rule* is of the form $u \rightarrow v$, where u is a string over O and v is a string over O_{tar}, where $O_{tar} = O \times TAR$ with $TAR = \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$;
5. $i_o \in \{1, 2, \ldots, m\}$ is a label of an elementary membrane (the *output membrane*)

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Maximal Parallelism

Both maximal parallelism (typical of molecular computations) and non determinism as standard strategy of rules application

Example: let the initial configuration be $[_3\ [_1\ ]_1\ [_2\ ]_2 a^5 b^3]_3$, over the alphabet $\Sigma = \{a, b, c\}$, and in membrane 3 let the rules be: $r : aab \rightarrow a_{in} bcc_{out}$ (where b is a catalyst) and $s : bb \rightarrow aac$

One may apply: r, s in parallel, or r twice, or s once, and the choice is done non-deterministically between the first two:

1. $[_3\ [_1\ ]_1\ [_2\ ]_2 a^5 b^3]_3 \Rightarrow [_3\ [_1 a]_1\ [_2\ ]_2 a^5 bc^2]c, [_3 [_1\ ]_1\ [_2 a]_2 a^5 bc^2]_3 c$
2. $[_3\ [_1\ ]_1\ [_2\ ]_2 a^5 b^3]_3 \Rightarrow$
   $[_3\ [_1 a^2]_1\ [_2\ ]_2 b^3 c]_3 c,\ [_3\ [_1 a]_1\ [_2 a]_2 b^3 c]_3 c,\ [_3\ [_1\ ]_1\ [_2 a^2]_2 b^3 c]_3 c$
3. $[_3\ [_1\ ]_1\ [_2\ ]_2 a^5 b^3]_3 \Rightarrow [_3\ [_1\ ]_1\ [_2\ ]_2 a^7 b^3 c]_3$

Symbol-object P systems – Configuration (transitions)

Definition 2

$$\Pi = (O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o),$$

- $C_0 = (\omega_1, \ldots, \omega_m)$ - initial configuration.
- $C = (\omega'_1, \ldots, \omega'_m)$ - configuration.
- Configuration transition: let $C' = (\omega'_1, \ldots, \omega'_m)$ and $C'' = (\omega''_1, \ldots, \omega''_m)$ be two configurations. $C' \Rightarrow C''$ (a *transition* occurs from C' to C''), if we can pass from C' to C'' by using the evolution rules from $R_1, \ldots, R_m$ in a non-deterministic maximal parallel manner.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Symbol-object P systems – Computation

Definition 3

$$\Pi = (O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o),$$

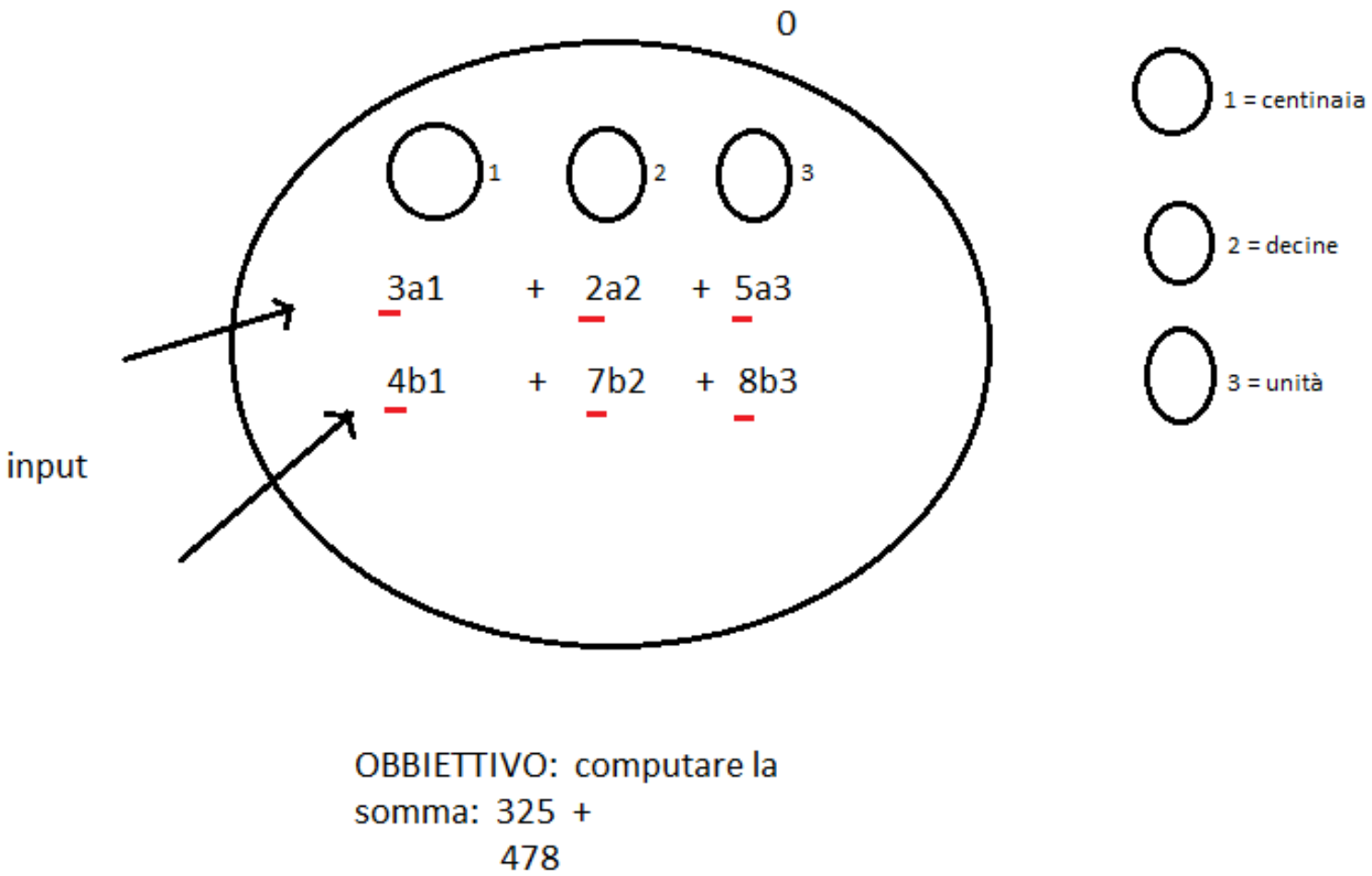
- Computation: sequence of configurations starting from the initial configuration.
- Successful computation: halting computation.
- Result of computation: the number of (vectors) of objects in the output region at halting; or, the number of objects that leave the system in a successful computation.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

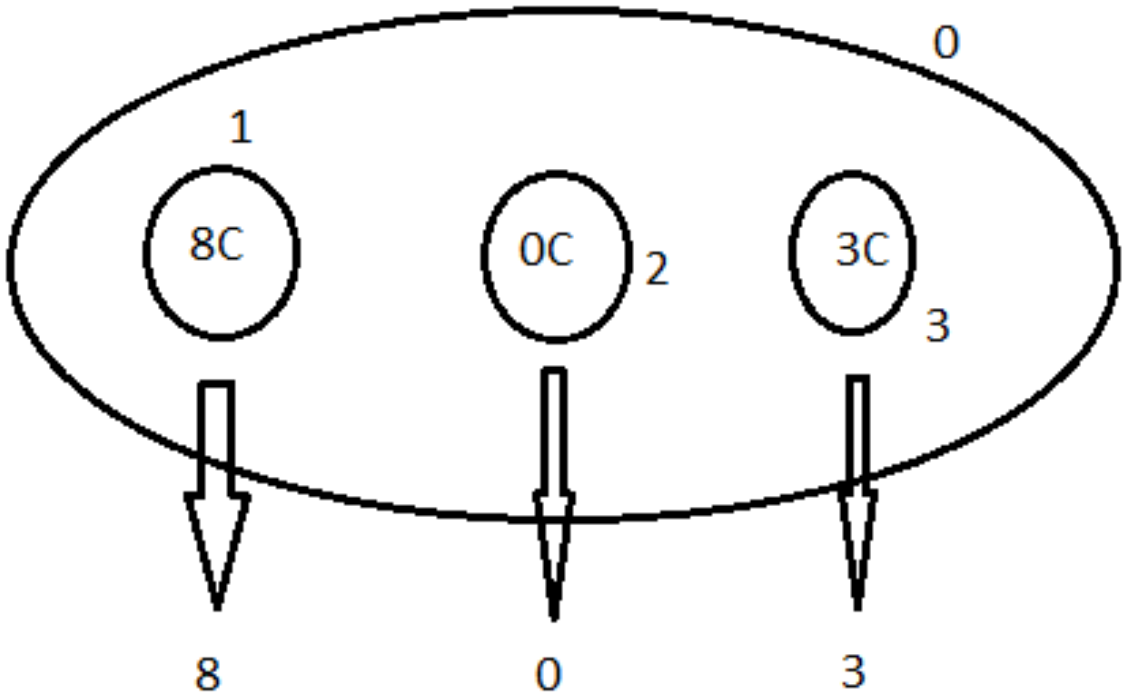
Exercise in Fig 2.47

Initial configuration $[_0\ [_1\ ]_1\ [_2\ ]_2\ [_3\ ]_3\ ]_0$ Numbers encoded respect. with a_1 hundreds, a_2 tens, and a_3 units, and b_1 hundreds, b_2 tens, and b_3 units.

$R_1 : [_0\ a_1\ ]_0 \rightarrow [_1\ c\ ]_1$
$R_2 : [_0\ a_2\ ]_0 \rightarrow [_2\ c\ ]_2$
$R_3 : [_0\ a_3\ ]_0 \rightarrow [_3\ c\ ]_3$
$R_4 : [_0\ b_1\ ]_0 \rightarrow [_1\ c\ ]_1$
$R_5 : [_0\ b_2\ ]_0 \rightarrow [_2\ c\ ]_2$
$R_6 : [_0\ b_3\ ]_0 \rightarrow [_3\ c\ ]_3$
$R_7 : [_3\ 10c\ ]_3 \rightarrow [_2\ c\ ]_2$
$R_8 : [_2\ 10c\ ]_2 \rightarrow [_1\ c\ ]_1$



Courtesy of Alessandro Mainente, Univ. VR, IT



CONTRO PROVA:

325+
478=

803

Courtesy of Alessandro Mainente, Univ. VR, IT

Symbol-object P systems – An example

Example 4 (Păun, 2002)

Let $\Pi_1 = (O, \mu, \omega_1, \omega_2, R_1, R_2, i_o)$ be a (cooperating) P system with the following components:

- $O = \{a, b, c\}$,
- $\mu = [_1\, [_2\,]_2]_1$,
- $\omega_1 = a^2$,
- $\omega_2 = \lambda$,
- $R_1 = \{a \to a(b, in_2)(c, in_2)^2, a^2 \to (a, out)^2\}$,
- $R_2 = \emptyset$,
- $i_o = 2$.

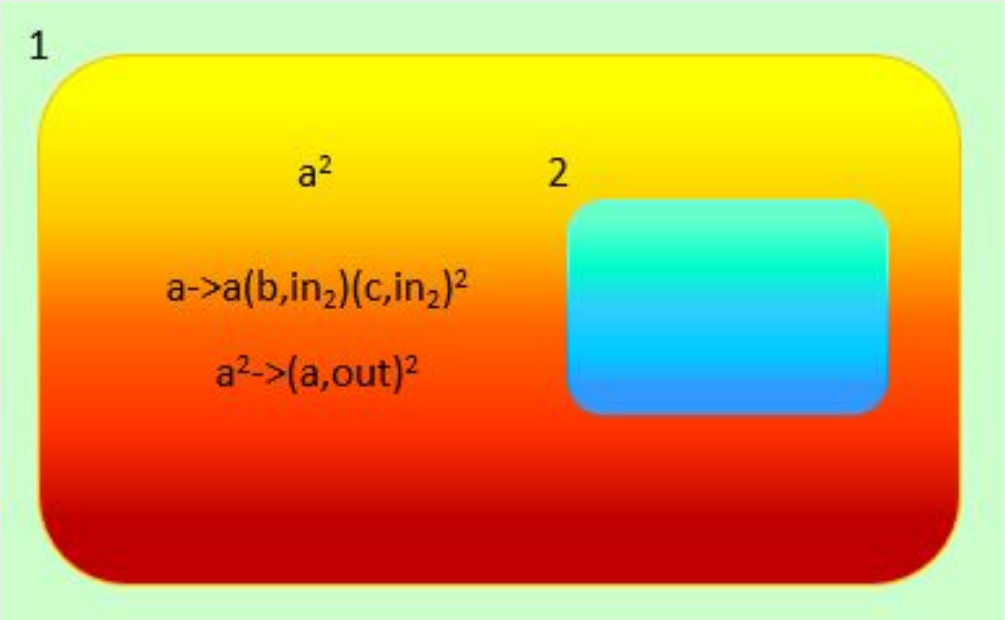
$N(\Pi_1) =$ (the computed set of numbers)???

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Symbol-object P systems – An example

Example 4 (Păun, 2002)

The initial configuration of Π_1:



Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Symbol-object P systems – An example

Example 4 (Păun, 2002)

- We start with two copies of a in membrane 1.
- In any subsequent configurations, except for the halting one, we will have two copies of a in membrane 1.
- If one of the copies is processed by rule $a \to a(b, in_2)(c, in_2)^2$, then the other one should also be processed by the same rule due to the maximal parallelism.
- If this rule is used, then a is reproduced in region 1 ($(a, here)$ indicates this fact, but *here* is omitted from the notation), and at the same time, one copy of b and two copies of c will be sent to membrane 2, which is the output membrane. There is no available rule in membrane 2.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Result Example 4

$$N(\Pi_1) = \{6n \mid n \geq 0\}$$

The process continues until both copies of a are sent outside the system through the application of rule $a^2 \rightarrow (a, out)^2$, thus the computation halts.

After $n + 1$ steps, for some n (it takes one step to send out object a), we will have object $b^{2n}c^{4n}$, which implies the output.

Not halting computations for a single (or an odd number of) a and an increasing amount of bcc in membrane 2. For an even number of a, say 4, we have 12-multiples (4n+8n), plus 6-multiples from halting computation with non massive application of the rule $a^2 \rightarrow (a, out)^2$. Input a^6 produces multiples of 18 (6+12), however included in 6-multiples.

Symbol-object P systems – Another example

Example 5 (Păun, 2002)

Let $\Pi_2 = (O, \mu, \omega, R_1, i_o)$ be a (non-cooperating) P system with the following components:

- $O = \{a, b\}$,
- $\mu = [_1\,]_1$,
- $\omega = a$,
- $R_1 = \{a \to ab^6, a \to (a, out)\}$,
- $i_o = 1$.

$N(\Pi_2) =$??? (Hint: $N(\Pi_2)$ is expected to be the same)

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities and dissolution

Priorities – deciding whether k divides n

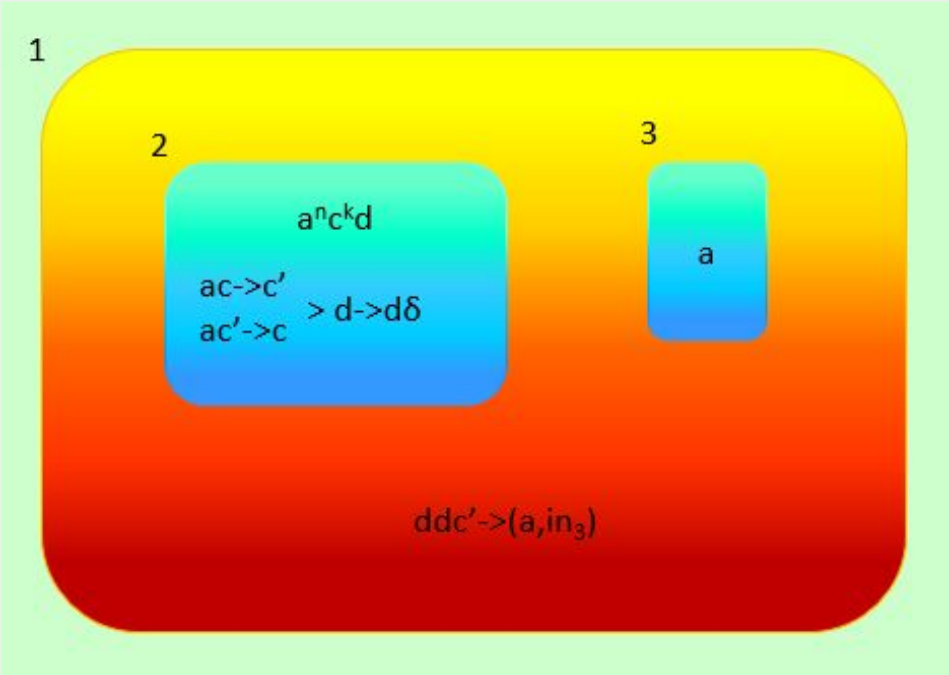
Example 8 (Gh. Păun, 2000)

Let $\Pi_5 = (O, \mu, \lambda, a^n c^k d, a, (R_1, \emptyset), (R_2, \rho_2), (\emptyset, \emptyset), 3)$ be a P system with priorities, where

- $O = \{a, c, c', d\}$,
- $\mu = [_1 [_2]_2 [_3]_3]_1$,
- $R_1 = \{dcc' \to (a, in_3)\}$,
- $R_2 = \{r_1 : ac \to c', r_2 : ac' \to c, r_3 : d \to d\delta\}$, $\rho_2 = \{r_1 > r_3, r_2 > r_3\}$.

Priorities – deciding whether k divides n

Example 8 (Gh. Păun, 2000)



Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities – deciding whether k divides n

Example 8 (Gh. Păun, 2000)

- We substract repeatedly k from n through rules $ac \to c'$ and $ac' \to c$: at each step k copies of a disappear, whereas c is reproduced (primed or non-primed versions of c will be produced alternatively).
- Rules $ac \to c'$ and $ac' \to c$ have priority over $d \to d\delta$.
- It means that we can dissolve membrane 2, only after we have exhausted n occurrences of a.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities – deciding whether k divides n

Example 8 (Gh. Păun, 2000)

- If n is a multiple of k (and only in this case), we never have both occurrences of c and c' simultaneously in membrane 2 (or in membrane 1, after the dissolution of membrane 2).
- Therefore rule $dcc' \to (a, in_3)$ will be used in membrane 1 if and only if n is not a multiple of k.
- This rule can only be used at most once, since we have only one occurrence of d.
- After using $dcc' \to (a, in_3)$, the computation stops.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Solution of Example 8

In conclusion, the computation always halts, and the output membrane contains two objects if and only if n is NOT a multiple of k.

In the opposite case, there remain only one object in the output membrane.

A control of type 0/1 objects in the output membrane would have been fine as well.

Priorities

Example 6 (Gh. Păun, 2000)

Let $\Pi_3 = (O, \mu, \omega_1, \omega_2, \omega_3, \omega_4, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), (R_4, \rho_4), i_o)$ be a P system with priorities, where

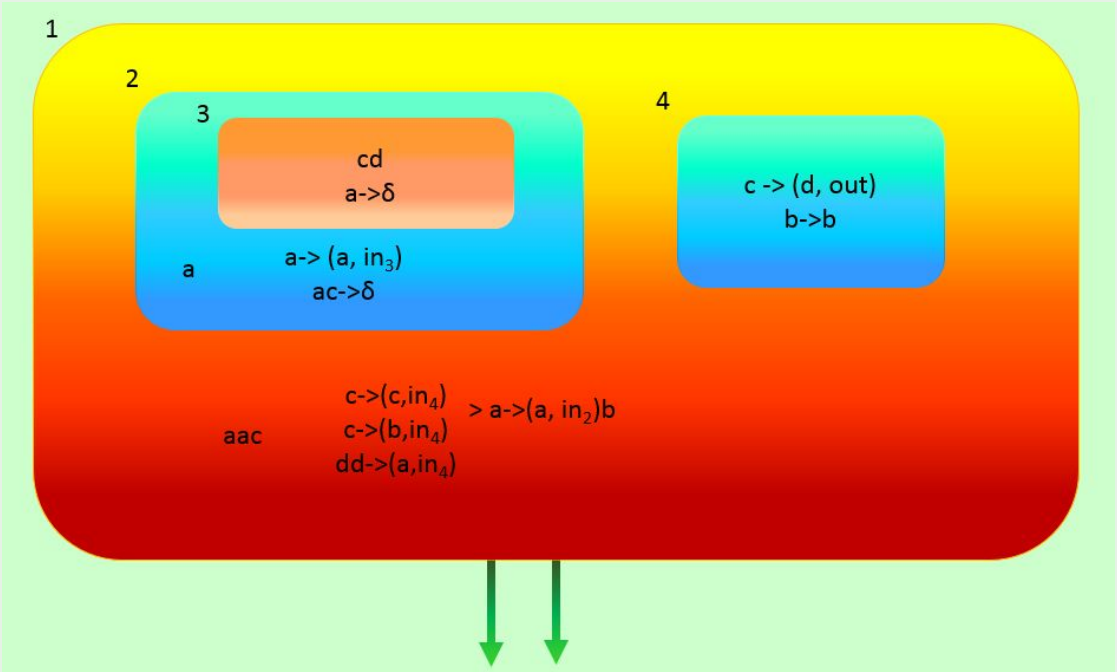
- $O = \{a, b, c, d\}$,
- $\mu = [_1\, [_2\, [_3\,]_3\,]_2\, [_4\,]_4\,]_1$,
- $\omega_1 = aac$, $R_1 = \{r_1 : c \to (c, in_4), r_2 : c \to (b, in_4), r_3 : a \to (a, in_2)b, dd \to (a, in_4)\}$, $\rho_1 = \{r_1 > r_3, r_2 > r_3\}$,
- $\omega_2 = a$, $R_2 = \{a \to (a, in_3), ac \to \delta\}$, $\rho_2 = \emptyset$,
- $\omega_3 = cd$, $R_3 = \{a \to \delta\}$, $\rho_3 = \emptyset$,
- $\omega_4 = \lambda$, $R_4 = \{c \to (d, out), b \to b\}$, $\rho_4 = \emptyset$,
- $i_o = 4$.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities

Example 6 (Gh. Păun, 2000)

Configuration transitions
STEP 0



Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities

Example 6 (Gh. Păun, 2000)

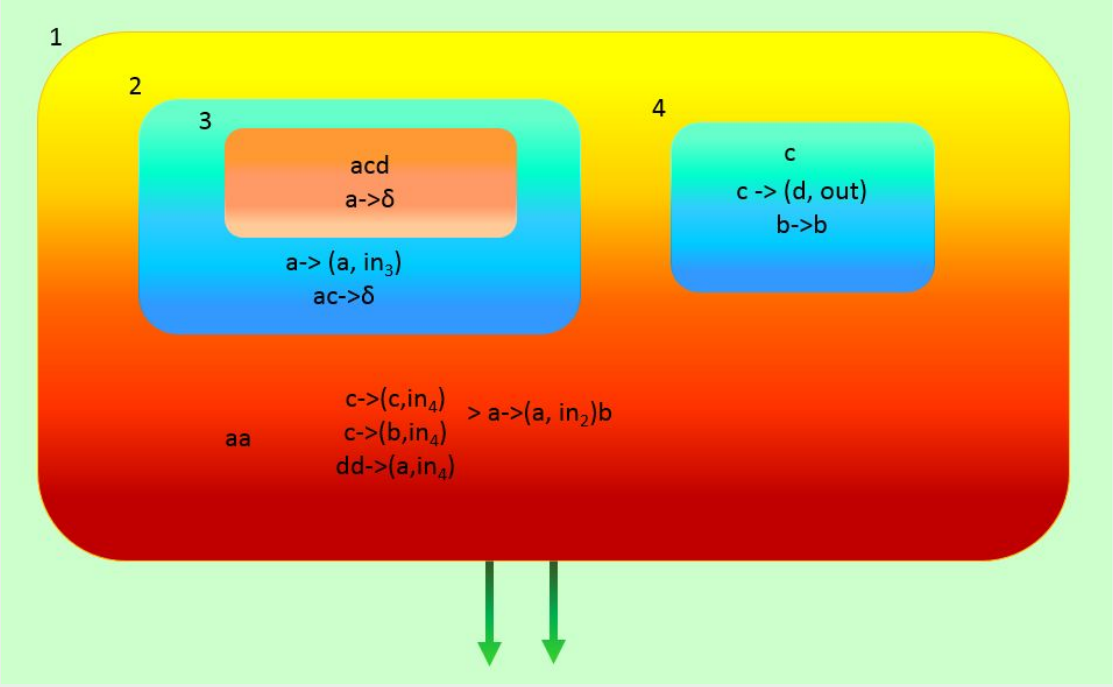
Step 0:

- In the initial configuration, we can apply one rule in membrane 1 and one in membrane 2.
- If we use rule $c \to (b, in_4)$ in membrane 1, then the computation will never halt (rule $b \to b$ will be used forever in membrane 4).
- Therefore we will not use rule $c \to (b, in_4)$, but rule $c \to (c, in_4)$.
- Since both these rules can be applied and have priority over rule $a \to (a, in_2)b$, this latter rule cannot be used.
- As a consequence, a symbol c will be sent from membrane 1 to membrane 4 and at the same time a symbol a from membrane 2 to membrane 3.

Priorities

Example 6 (Gh. Păun, 2000)

STEP 1



Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities

Example 6 (Gh. Păun, 2000)

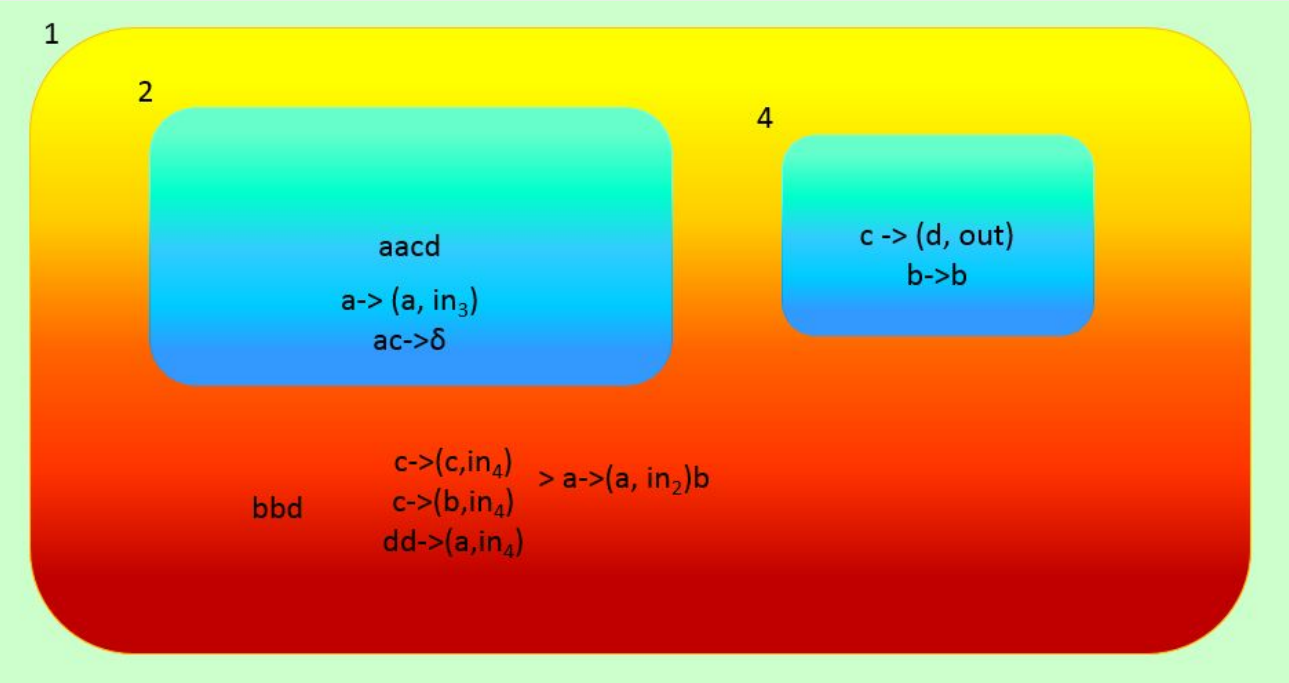
Step 1:

- At this step, neither rule $c \to (b, in_4)$, nor rule $c \to (c, in_4)$ can be applied in membrane 1.
- It means that rule $a \to (a, in_2)b$ has to be used for both copies of a present in membrane 1.
- At the same time, rule $a \to \delta$ will be used in membrane 3, dissolving it, and rule $c \to (d, out)$ will be used in membrane 4, sending a copy of d to membrane 1.
- As a result, membrane 1 will contain the multiset bbd, membrane 2 $aacd$, membrane 4 is empty and membrane 3 does no longer exist (hence rule $a \to (a, in_3)$ becomes useless in membrane 2).

Priorities

Example 6 (Gh. Păun, 2000)

STEP 2



Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities

Example 6 (Gh. Păun, 2000)

Step 2:

- At this step, first rule $ac \to \delta$ will be used in membrane 2, dissolving it and releasing the remaining objects ad.
- Therefore membrane 1 will contain the multiset $abbdd$, which makes it possible for the first time to use rule $dd \to (a, in_4)$ from membrane 1.
- It consumes two copies of d and sends one copy of a to membrane 4.
- At the same time, rule $a \to (a, in_2)b$ will send one copy of a to membrane 2.
- Afterwards, no further rule can be applied.

Priorities – generating square numbers

Example 7 (Gh. Păun, 2000)

Let $\Pi_4 = (O, \mu, \omega_1, \omega_2, \omega_3, \omega_4, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), (R_4, \rho_4), i_o)$ be a P system with priorities, where

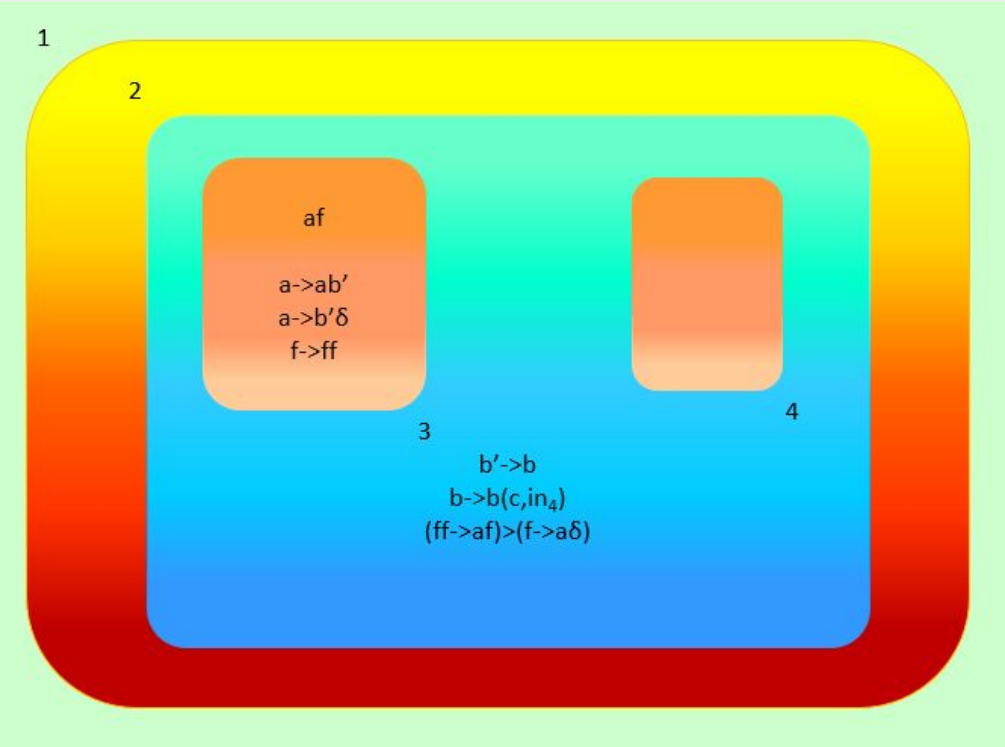
- $O = \{a, b, b', c, f\}$,
- $\mu = [_1 [_2 [_3]_3 [_4]_4]_2]_1$,
- $\omega_1 = \lambda$, $R_1 = \emptyset$, $\rho_1 = \emptyset$,
- $\omega_2 = \lambda$, $R_2 = \{b' \to b, b \to b(c, in_4), r_1 : ff \to af, r_2 : f \to a\delta\}$, $\rho_2 = \{r_1 > r_2\}$,
- $\omega_3 = af$, $R_3 = \{a \to ab', a \to b'\delta, f \to ff\}$, $\rho_3 = \emptyset$,
- $\omega_4 = \lambda$, $R_4 = \emptyset$, $\rho_4 = \emptyset$,
- $i_o = 4$.

$N(\Pi_4)$= ???

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities – generating square numbers

Example 7 (Gh. Păun, 2000)



Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Priorities – generating square numbers

Example 7 (Gh. Păun, 2000)

- There is not any applicable rule in membrane 2 owing to the absence of any objects.
- The only possibility is to start with membrane 3, using objects a and f, present each copy in one.
- Using rules $a \to ab'$ and $f \to ff$ for all currently available occurrences of a and f, after n steps, we obtain n occurrences of b' and 2^n occurrences of f.

Priorities – generating square numbers

Example 7 (Gh. Păun, 2000)

- At any step, instead of applying rule $a \to ab'$, rule $a \to b'\delta$ can also be used.
- Observe that we always have one copy of a only.
- After the applying rule $a \to b'\delta$, we have $n+1$ occurrences of b' and 2^{n+1} occurrences of f and membrane 3 will be dissolved.
- The obtained configuration is $[_1\, [_2\, b'^{n+1} f^{2^{n+1}}, b' \to b, b \to b(c, in_4), r_1 : ff \to af, r_2 : f \to a\delta, r_1 > r_2, [_4\,]_4\,]_2\,]_1$.

Priorities – generating square numbers

Example 7 (Gh. Păun, 2000)

- The rules of the former active membrane are discarded, the rules of membrane 2 are now active.
- Due to the priority relation, we have to use rule $ff \rightarrow af$ as long as possible.
- In one step b'^{n+1} will be transformed into b^{n+1} and the number of occurrences of f will be divided by 2.
- In the next step, each occurrence of b will introduce one occurrence of c in membrane 4 ($n+1$ cs altogether).
- At the same time, the number of occurrences of f will again be divided by 2.

Priorities – generating square numbers

Example 7 (Gh. Păun, 2000)

- Continuing the computations, we may see that at each step further $n+1$ occurrences of c are introduced in the output membrane.
- This can be done for $n+1$ steps: n times when rule $ff \to af$ is used (until there is only one f) and once when using rule $f \to a\delta$.
- At his step, membrane 2 will be dissolved and its rules will be removed.
- No further steps are possible.
- We reach configuration is $[_1\, a^{2^{n+1}}\, b^{n+1}, [_4\, c^{(n+1)^2}\,]_4\,]_1$.
- Consequently, $N(\Pi_4) = \{m^2 \mid m \geq 1\}$.

Courtesy of Dr Katalin Lázár, Eötvs Loránd University, Budapest, Hungary

Homework: P system computing a function of n

Input skin membrane $[_1\ ]_1$. Encoding for n: a^n.

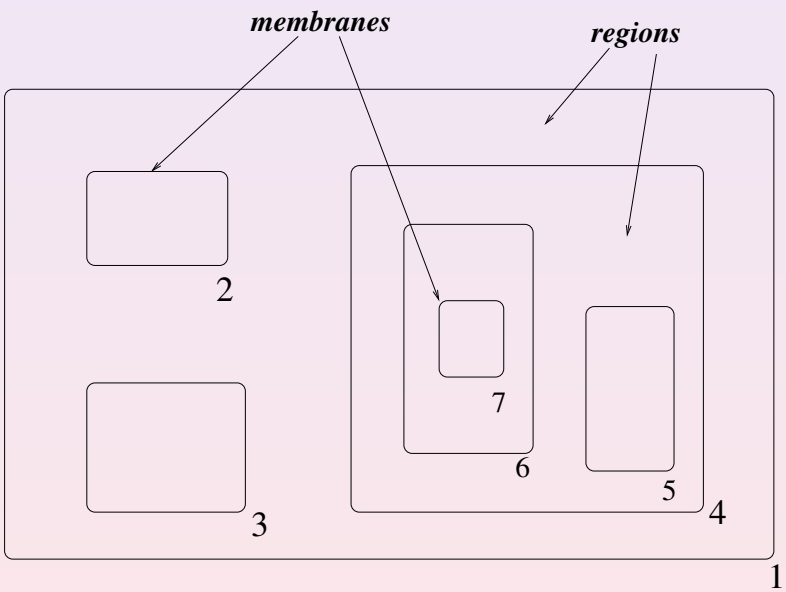
Initial configuration: $[_1\ c\ [_2\ ]_2\ ]_1$.

Rules: $R_1 = \{a \to b_1 b_2,\ cb_1 \to cb_1',\ b_2 \to b_2 e_{in}\ |_{b_1}\}$

Output membrane: $[_2\ ]_2$. (Output is the content of membrane 2 after computation halts).

(Membrane) P systems

Distributed computational model inspired by the structure and the functioning of the living cell



- Multisets of objects
- Localized/global evolution rules
- Synchronization
- Non-determinism and maximal parallelism
- Transitions on system configurations

- Languages-generating machines
- Models for biochemical/cell processes.

Natural Computing Paradigm versus Turing Paradigm

- What do biological systems compute?
- A Turing machine, given an input, goes on until it reaches a final state where the output is encoded
- A biological system is a "non terminating machine" (when termination arises life ends) where at any time many cooperating processes are running.

4

Courtesy of prof. Manca, Univ. VR, IT

Issues about P Systems

Membrane systems as realistic (to model biological processes), powerful (in the Turing hierarchy), efficient (capable to solve NP complete problems), elegant (under some notions of complexity) as possible.

Ex: $NP_2(Pri, Cat)$, $NP_4(Cat, \delta, \tau)$, $NP_1(Cat_2)$, $NP_3(Syn_1, Ant_1)$ are all universal systems.

A more recent perspective

A natural perspective considers such systems as dynamical systems. Thus, the interest is moved from their halting configurations, to the orbits, the cycles, the fixed points of their computational dynamics, while the states space being the set of possible configurations.

Biological dynamics

Biological dynamics time evolution of a set of *relevant* variables

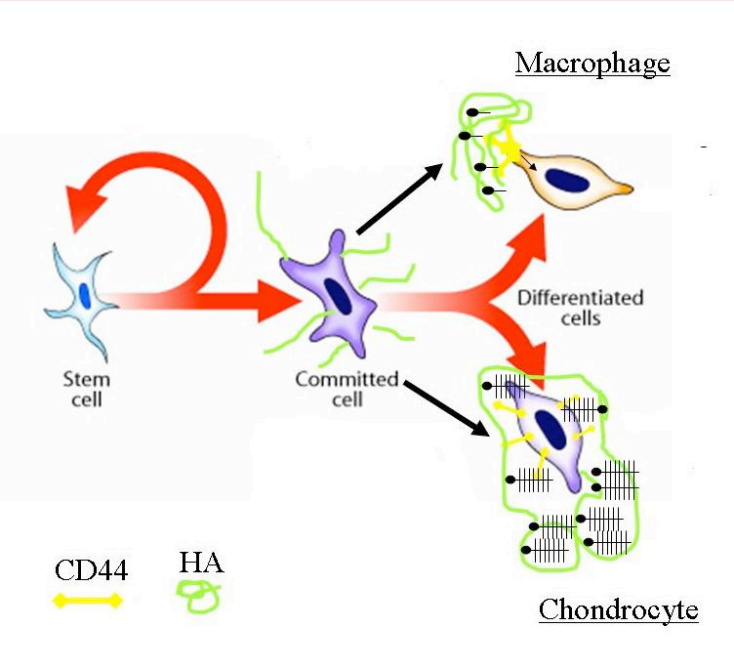
$$x_1(t), \ldots, x_n(t)$$

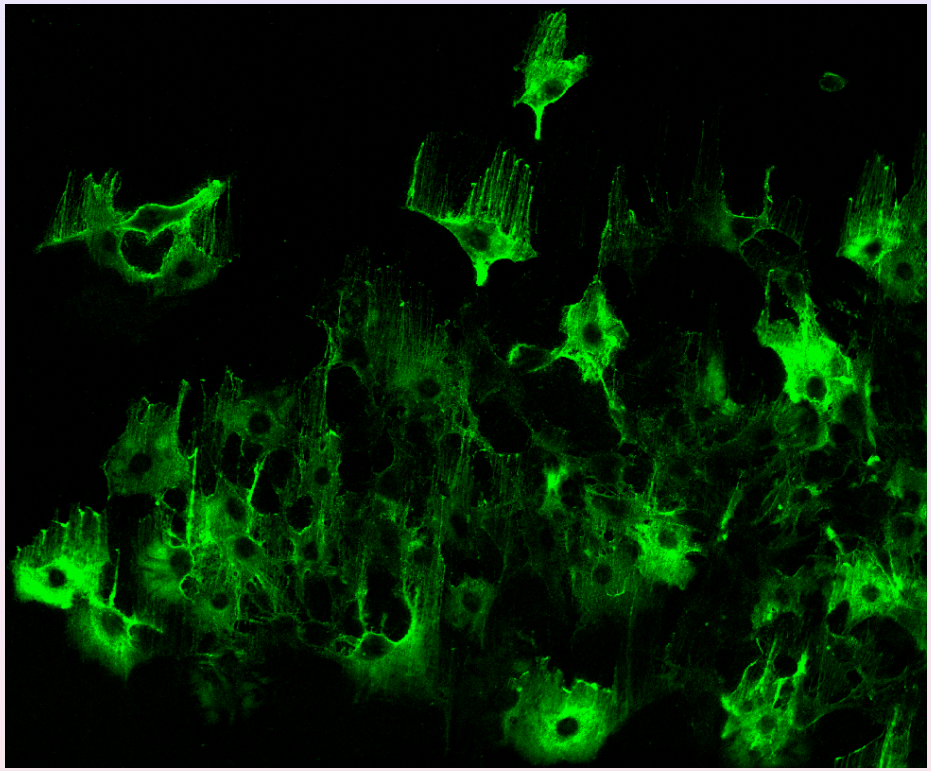
Actual versus *observed* dynamics, known at a qualitative or quantitative level (measures: numerical time series).

Modeling *complex* biological systems promotes the analysis of both the biological phenomenon and the mathematical model:

- $\rightarrow$ understanding the functioning of biological dynamics;
- $\rightarrow$ developing mathematical theory for reliable models.

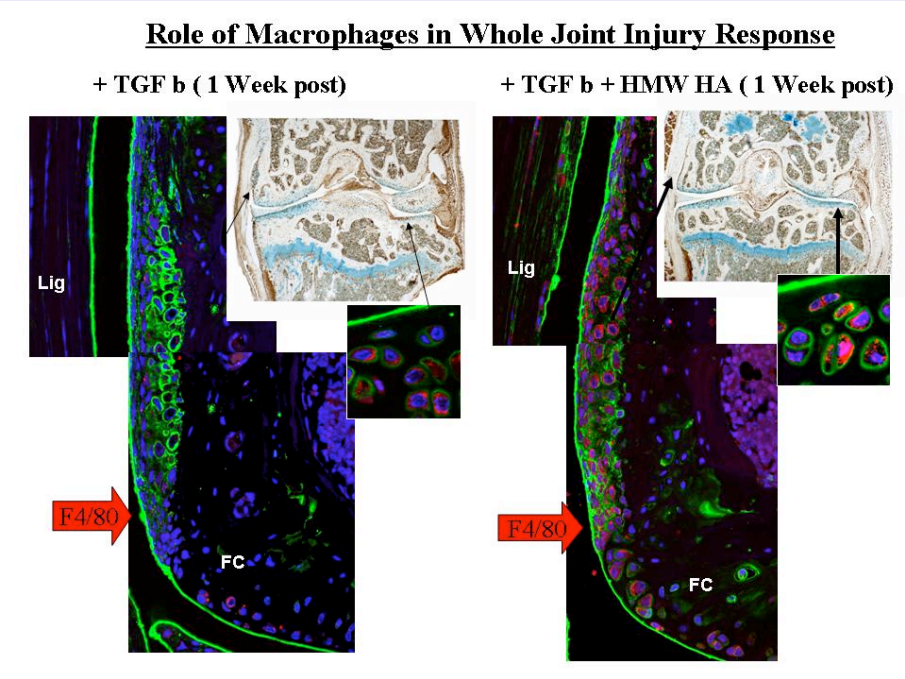
Case study: healing of knee joint injury

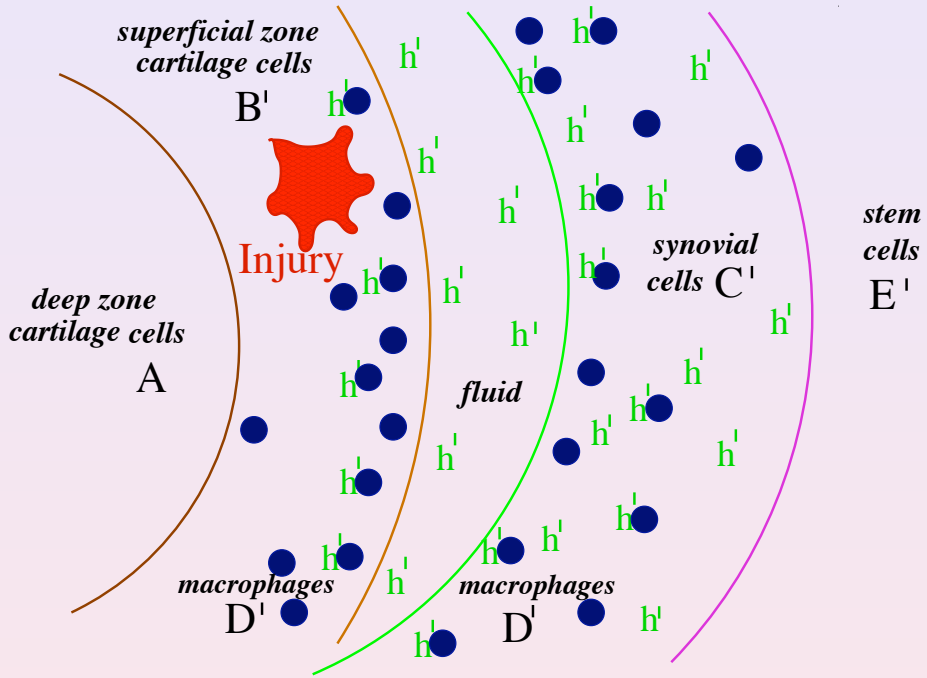




G. F., N. Jonoska, B. Osborn, A. Plaas, *Knee joint injury and repair modeled by membrane systems*, BioSystems 91, pp 473–488, 2008.

Modeling process



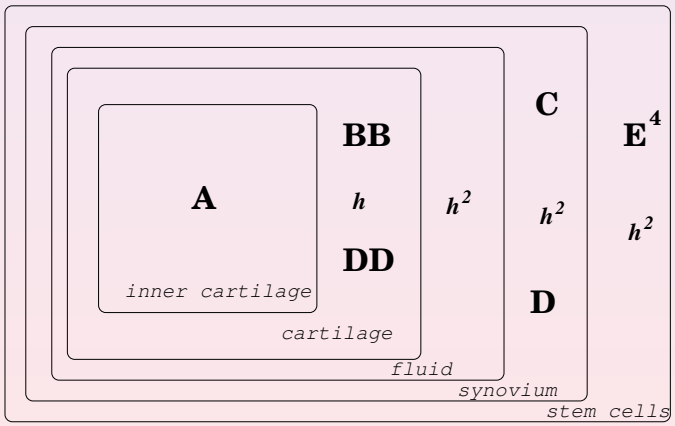


G. F., N. Jonoska, B. Osborn, A. Plaas, *Knee joint injury and repair modeled by membrane systems*, BioSystems 91, pp 473–488, 2008.

Knee Joint Injury Modeled by P Systems

A P system to model actual biological dynamics.

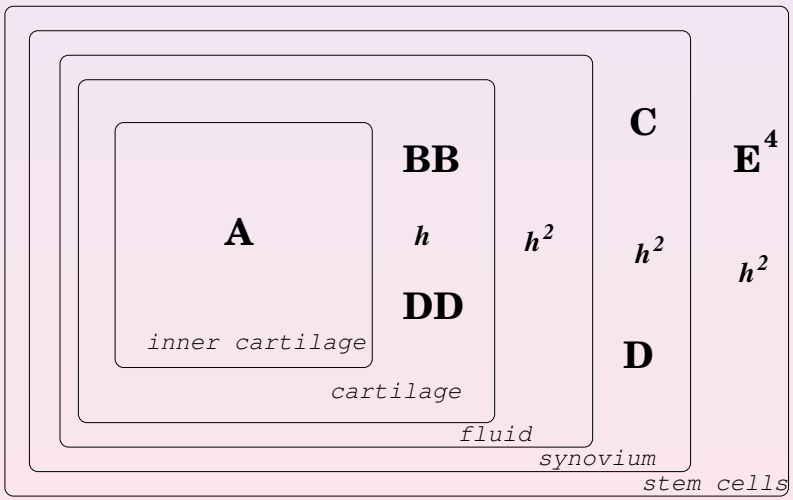
$r_1 : B \to BB\ h\ h_{out}$
$r_2 : C \to C\ h_{in}$
$r_3 : h \to h_{out}$
$r_4 : Bh \to D$
$r_5 : DD \to D$



$$[[[[[\ A\ ]\ BD\ ]\quad]\ CD\ ]\ E^4\ ]$$
$$\downarrow 1,2$$
$$[[[[[\ A\ ]\ BBhD\ ]\ h^2\ ]\ CD\ ]\ E^4\ ]$$
$$\downarrow 1,2,3,4$$
$$[[[[[\ A\ ]\ BBhDD\ ]\ h^2\ ]\ h^2 CD\ ]\ E^4\ ]$$
$$\downarrow 1,2,3,4,5$$
$$[[[[[\ A\ ]\ BBhDD\ ]\ h^2\ ]\ h^2 CD\ ]\ h^2 E^4\ ]$$
$$\downarrow 1,2,3,4,5$$
$$[[[[[\ A\ ]\ BBhDD\ ]\ h^2\ ]\ h^2 CD\ ]\ h^2 E^4\ ]$$
$$\vdots$$

Healing of knee joint injury modeled by a P system

$r_1 : B \to BB\ h\ h_{out}$
$r_2 : C \to C\ h_{in}$
$r_3 : h \to h_{out}$
$r_4 : Bh \to D$
$r_5 : DD \to D$

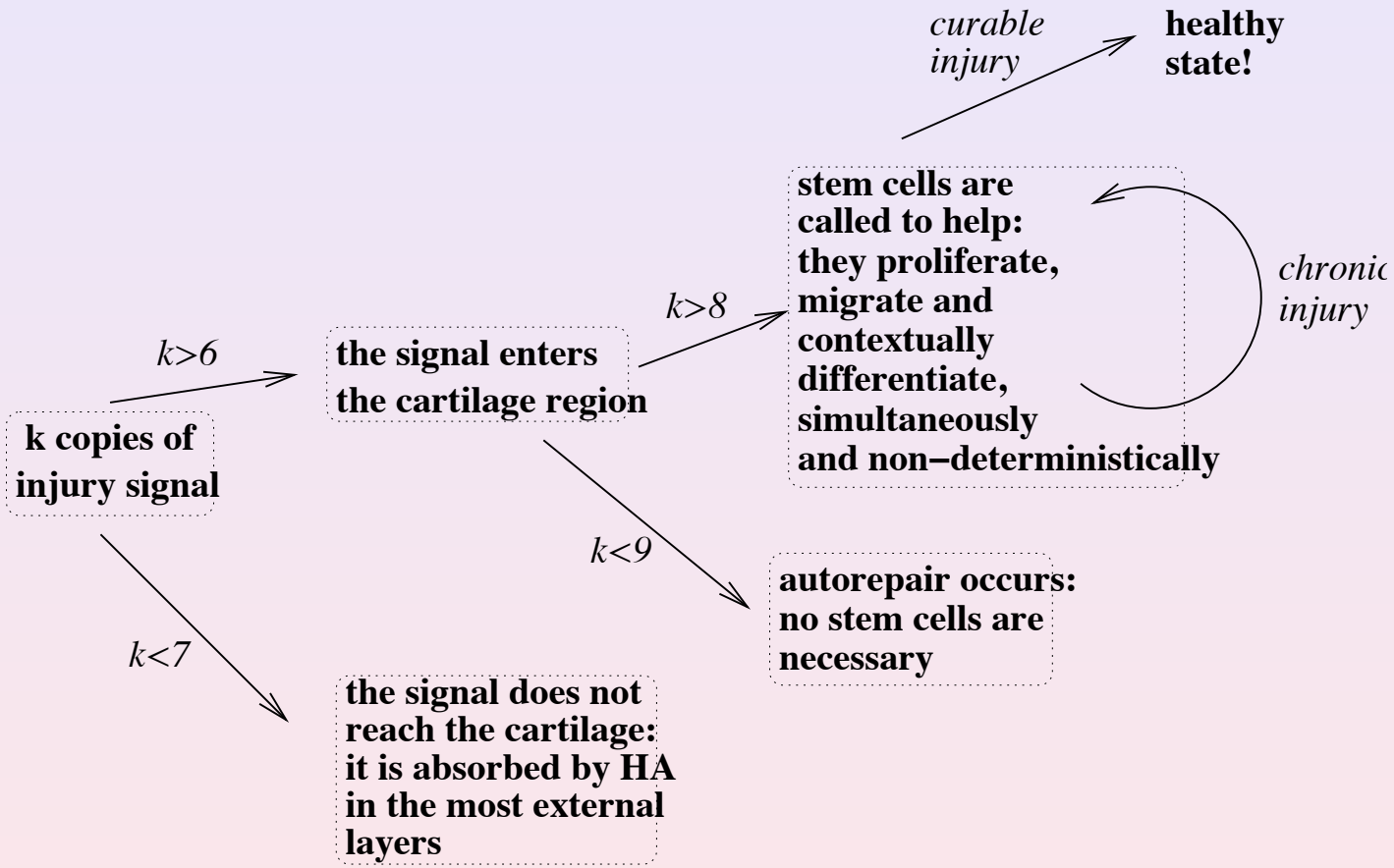


$$[[[[[\ A\ ]\ BD\ ]\quad]\ CD\ ]\ E^4\ ]$$
$$\downarrow_{1,2}$$
$$[[[[[\ A\ ]\ BBhD\ ]\ h^2\ ]\ CD\ ]\ E^4\ ]$$
$$\downarrow_{1,2,3,4}$$
$$[[[[[\ A\ ]\ BBhDD\ ]\ h^2\ ]\ h^2 CD\ ]\ E^4\ ]$$
$$\downarrow_{1,2,3,4,5}$$
$$[[[[[\ A\ ]\ BBhDD\ ]\ h^2\ ]\ h^2 CD\ ]\ h^2 E^4\ ]$$
$$\downarrow_{1,2,3,4,5}$$
$$[[[[[\ A\ ]\ BBhDD\ ]\ h^2\ ]\ h^2 CD\ ]\ h^2 E^4\ ]$$
$$\vdots$$

conditioned maximal parallelism

$\mathbf{S^k}$

Main (qualitative) result



Key role of cell differentiation from stem cells to macrophages in the intrinsic non-determinism of the healing process.

What kind of dynamics?

Let P system states be infinite, $X_i \in \mathbb{N}_n$ (vectors or matrices).
Let us focus on the transitions

$$X_i \rightarrow X_j$$

that we have if "the rules of the P system make it to pass from the configuration X_i to X_j".

Any function to represent the application of rewriting rules?
How to represent movements of objects between membranes?

It depends on the strategy. Two "deterministic" possibilities:

- Stochastic matrices (a probability of application is associated to each rule)
- Metabolic algorithm (a polynomial function) which in each step performs a distribution of objects among all the rules.

What kind of dynamics?

Let P system states be infinite, $X_i \in \mathbb{N}_n$ (vectors or matrices).
Let us focus on the transitions

$$X_i \rightarrow X_j$$

that we have if "the rules of the P system make it to pass from the configuration X_i to X_j".
Any function to represent the application of rewriting rules?
How to represent movements of objects between membranes?

It depends on the strategy. Two "deterministic" possibilities:

- Stochastic matrices (a probability of application is associated to each rule)
- Metabolic algorithm (a polynomial function) which in each step performs a distribution of objects among all the rules.

What kind of dynamics?

Let P system states be infinite, $X_i \in \mathbb{N}_n$ (vectors or matrices).
Let us focus on the transitions

$$X_i \rightarrow X_j$$

that we have if "the rules of the P system make it to pass from the configuration X_i to X_j".
Any function to represent the application of rewriting rules?
How to represent movements of objects between membranes?

It depends on the strategy. Two "deterministic" possibilities:

- Stochastic matrices (a probability of application is associated to each rule)
- Metabolic algorithm (a polynomial function) which in each step performs a distribution of objects among all the rules.

What kind of dynamics?

Let P system states be infinite, $X_i \in \mathbb{N}_n$ (vectors or matrices).
Let us focus on the transitions

$$X_i \rightarrow X_j$$

that we have if "the rules of the P system make it to pass from the configuration X_i to X_j".
Any function to represent the application of rewriting rules?
How to represent movements of objects between membranes?

It depends on the strategy. Two "deterministic" possibilities:

- Stochastic matrices (a probability of application is associated to each rule)
- Metabolic algorithm (a polynomial function) which in each step performs a distribution of objects among all the rules.

What kind of dynamics?

Let P system states be infinite, $X_i \in \mathbb{N}_n$ (vectors or matrices).
Let us focus on the transitions

$$X_i \to X_j$$

that we have if "the rules of the P system make it to pass from the configuration X_i to X_j".
Any function to represent the application of rewriting rules?
How to represent movements of objects between membranes?

It depends on the strategy. Two "deterministic" possibilities:

- Stochastic matrices (a probability of application is associated to each rule)
- Metabolic algorithm (a polynomial function) which in each step performs a distribution of objects among all the rules.

What about the non-deterministic system evolution?

$$r_1 : \lambda \to A \qquad \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -2 & -1 & 1 \\ 3 & 0 & -1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{pmatrix} A \\ B - A \\ -2A - B + C \\ 3A - C \end{pmatrix}$$

$$r_2 : A \to B$$

$$r_3 : AAB \to C$$

$$r_4 : C \to AAA$$

$$[A^2B] \Rightarrow_{1,2} [A^2B^2] \qquad \begin{pmatrix} 2 + x_1 \\ 1 + x_2 \\ 0 + x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}, \quad \text{or} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\Downarrow_{1,3}$$

$$[AC]$$

$A - A + B - 2A - B + C + 3A - C = A$. It does not work, and some constraints are necessary. For example:

- applicability of each rule requires premises present in the state (first 3 rules): still $A - A + B - 2A - B + C = -2A + C$.
- interconnectivity of rules (either r_2 or r_3 can be applied): $A - A + B = B$ or $A - 2A - B + C = -A - B + C$!

Open Problems (CiE, 2007)

- Can the non-deterministic evolution of membrane systems be described by linear operators?

  If configurations are described by matrices, in order to express non-determinism the (partial) operator of the P dynamical system should be a set of (applicable) matrices.

- How can be represented, in terms of system dynamics, the moving of objects among membranes? How studying the dynamics on the membrane structure (a tree) of a P system having also dissolution/duplication rules?

Open Problems (CiE, 2007)

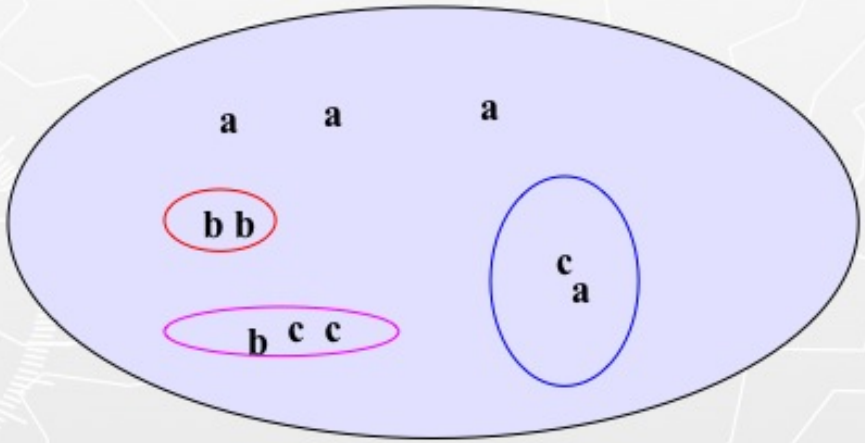
- Can the non-deterministic evolution of membrane systems be described by linear operators?

  If configurations are described by matrices, in order to express non-determinism the (partial) operator of the P dynamical system should be a set of (applicable) matrices.

- How can be represented, in terms of system dynamics, the moving of objects among membranes? How studying the dynamics on the membrane structure (a tree) of a P system having also dissolution/duplication rules?

Metabolic System

The **state** of a metabolic system is given by the **types**, the **localization**, and **quantity** of its metabolites.



P systems the natural theoretical framework

Courtesy of prof. Manca, Univ. VR, IT

Traditional modeling

ODE for Metabolism

Autonomous Differential Equations
for computing the dynamics
(a state is an n-dimension vector)

$$dx_1/dt = f_1(x_1, x_2, \ldots, x_n)$$
$$dx_2/dt = f_2(x_1, x_2, \ldots, x_n)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$dx_n/dt = f_n(x_1, x_2, \ldots, x_n)$$

Eberhard O. Voit Computational Analysis of Biochemical Systems, C.U.P, 2000 (+ PLAS software)

6

Courtesy of prof. Manca, Univ. VR, IT

Modeling and computation

Computational model: dynamical system, discrete in space and time. A computation as a sequence of states is a dynamics.

Metabolic P systems [Manca et al. '05] with algebraic and algorithmic procedures compute the dynamics of a reactor by a recurrence system.

New methodology. From time series[5] to an MP grammar which

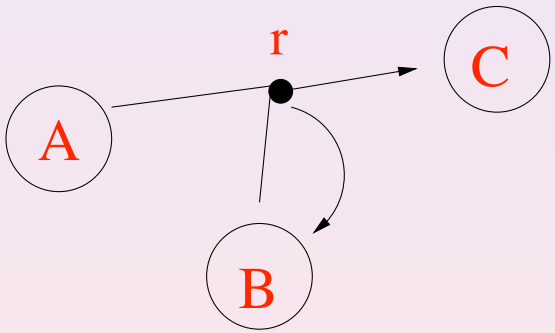
1. includes the biological knowledge of the phenomenon,
2. has a 'own' dynamics fitting the time series.

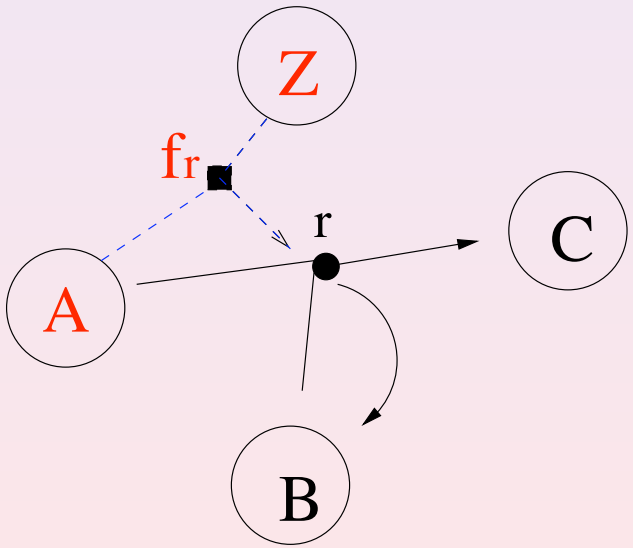
[5] quantities sperimentally measured at macroscopic temporal scales

Metabolic P systems: an intuition

Multiset rewriting system: a set of reactions r (transforming metabolites), corresponding regulators f_r, whose value (*flux* u_r) is the molar quantity of matter transformed by the reaction.

$A + B \xrightarrow{r} B + C$ and $f_r = 3az^2$, with $y = |Y|$, $Y = A, B, C, Z$





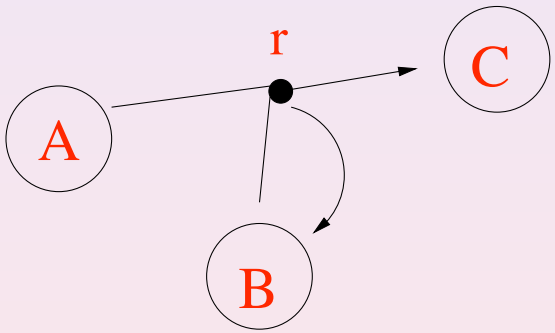
Z parameter
(massless entity)

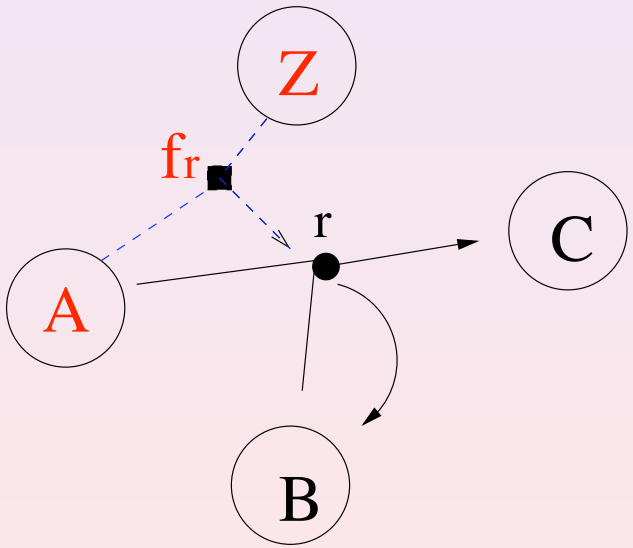
- $f_r = 1.3z^{-2}$ (inhibition)
- $f_r = 0.8a^3$ (promotion)
- $\lambda \xrightarrow{r} B,\ f_r = k$ (production)
- $C \xrightarrow{r} \lambda,\ f_r = k$ (degradation)

Metabolic P systems: an intuition

Multiset rewriting system: a set of reactions r (transforming metabolites), corresponding regulators f_r, whose value (*flux* u_r) is the molar quantity of matter transformed by the reaction.

$A + B \xrightarrow{r} B + C$ and $f_r = 3az^2$, with $y = |Y|$, $Y = A, B, C, Z$





Z parameter
(massless entity)

- $f_r = 1.3z^{-2}$ (inhibition)
- $f_r = 0.8a^3$ (promotion)
- $\lambda \xrightarrow{r} B,\ f_r = k$ (production)
- $C \xrightarrow{r} \lambda,\ f_r = k$ (degradation)

MP System

$$M = (X, V, R, Q, \nu, \mu, \tau, \sigma_0, \Phi, \delta)$$

X = Substances

V = Parameters $\{h_v \mid v \in V\}$ $h_v : N \dashrightarrow R$

R = Reactions

Q = States

A state q is a function

$q : X \cup V \dashrightarrow R$

ν = Mole size

$q = \{x_1, x_2, \ldots, v_1, v_2, \ldots\}$

μ = Mole mass

$q[i] = (x_1[i], x_2[i], \ldots, v_1[i], v_2[i], \ldots)$

τ = Time unit

σ_0 = Initial state

H = Parameter Functions

Φ= Flux Regulation Functions $\varphi_r : Q \dashrightarrow R$

δ = Dynamics

14

Courtesy of prof. Manca, Univ. VR, IT

Metabolic Algorithm

Dynamics

δ: N --> Q

$\delta(i) = (X[i] , V[i])$

$(X[0],V[0]) = \sigma_0$

Regulation

$\phi = \{\varphi_r \mid r \in R\}$

$U[i] = \phi(X[i],V[i])$

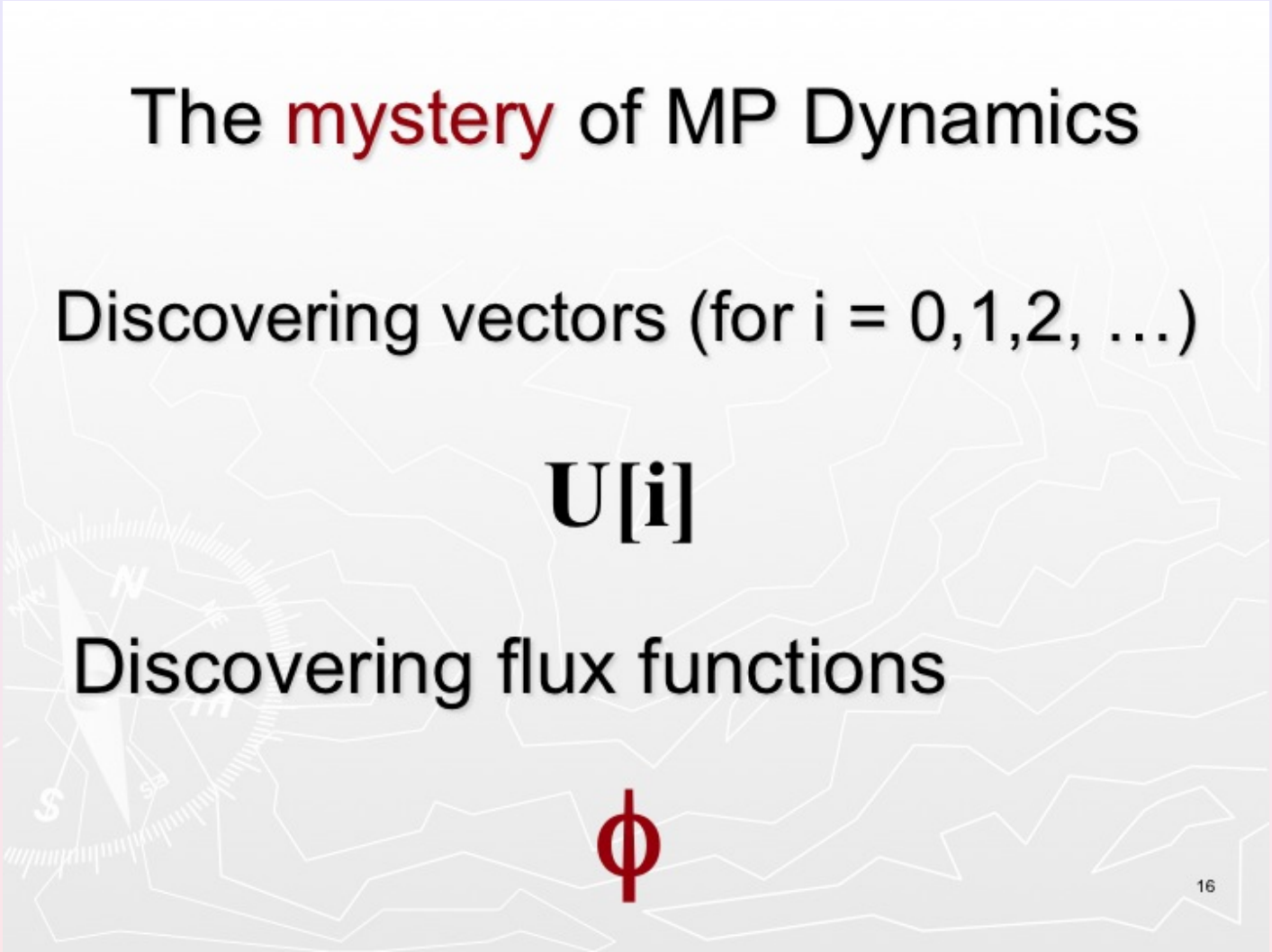
$$X[i+1] = A \times U[i] + X[i]$$

A =Stoichiometric Matrix, × = matrix product

Manca V., The Metabolic algorithm for P systems: Principles and Applications, TCS, 2008

15

Courtesy of prof. Manca, Univ. VR, IT



Courtesy of prof. Manca, Univ. VR, IT