Membrane Computing at (more than) Twelve Years

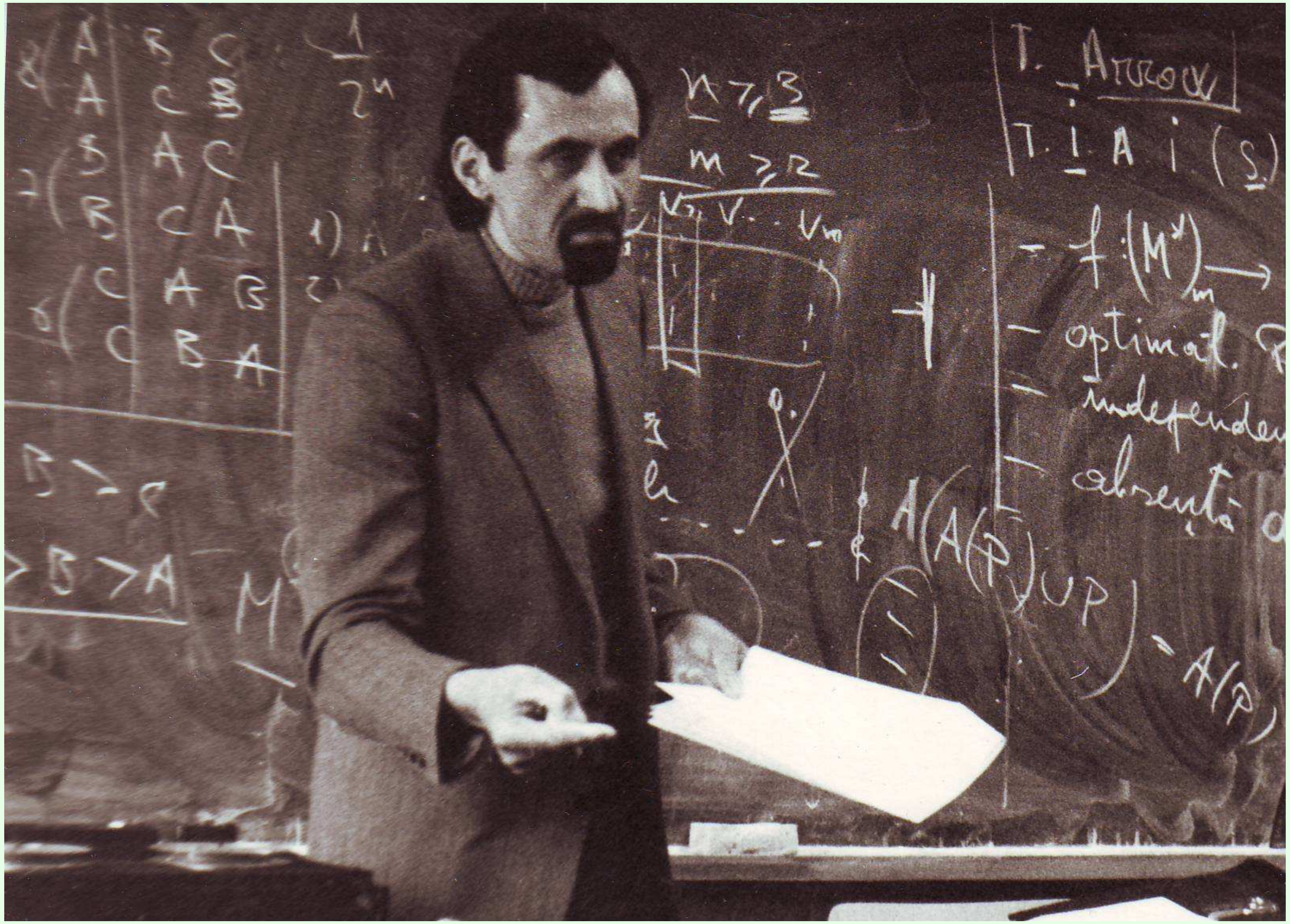
Gheorghe Păun
Romanian Academy, București,
RGNC, Sevilla University, Spain
george.paun@imar.ro, gpaun@us.es

Everything started 12 years ago, in Turku...

Great environment...

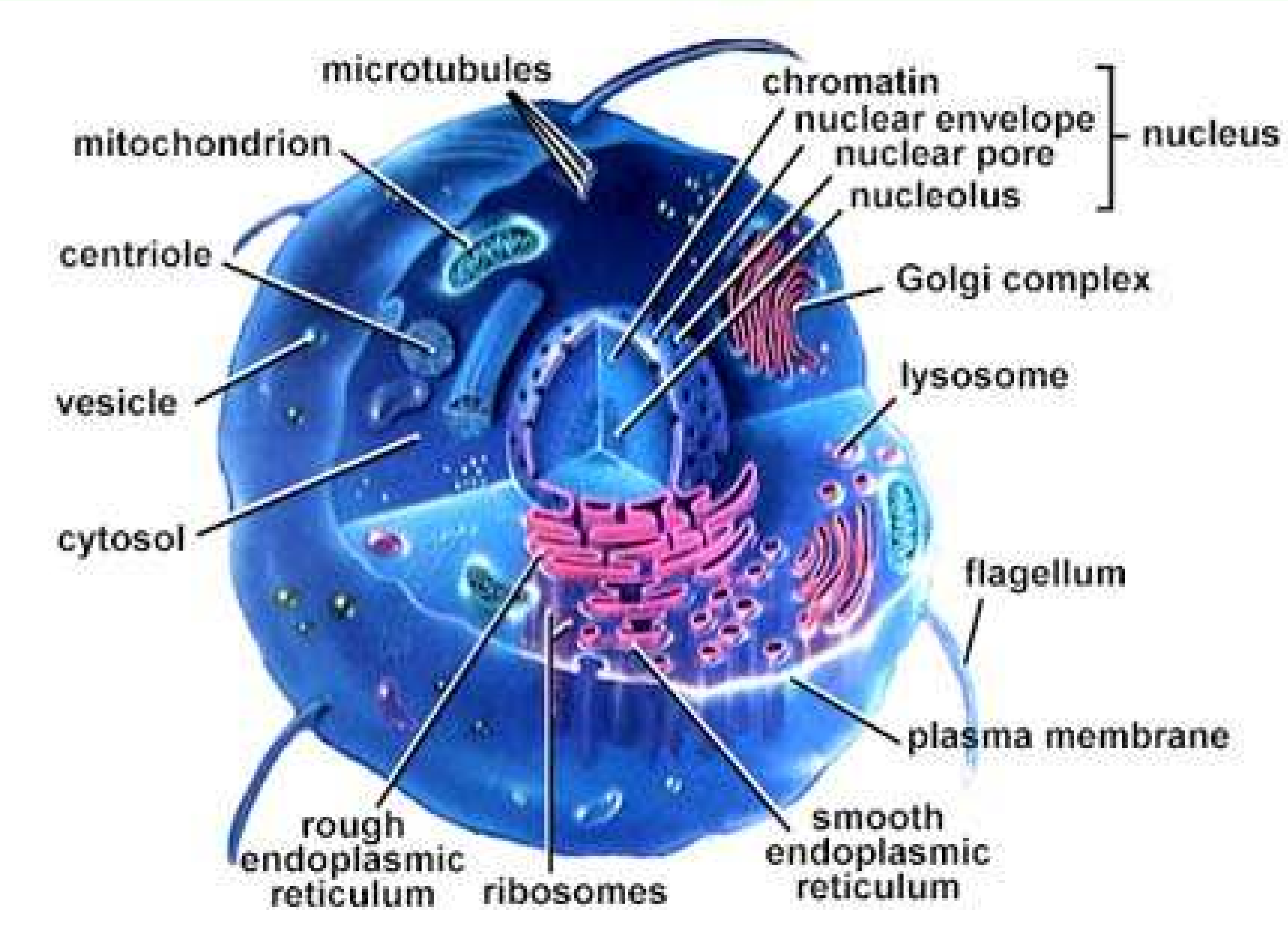
...with really big hats...

...also the Magician was around



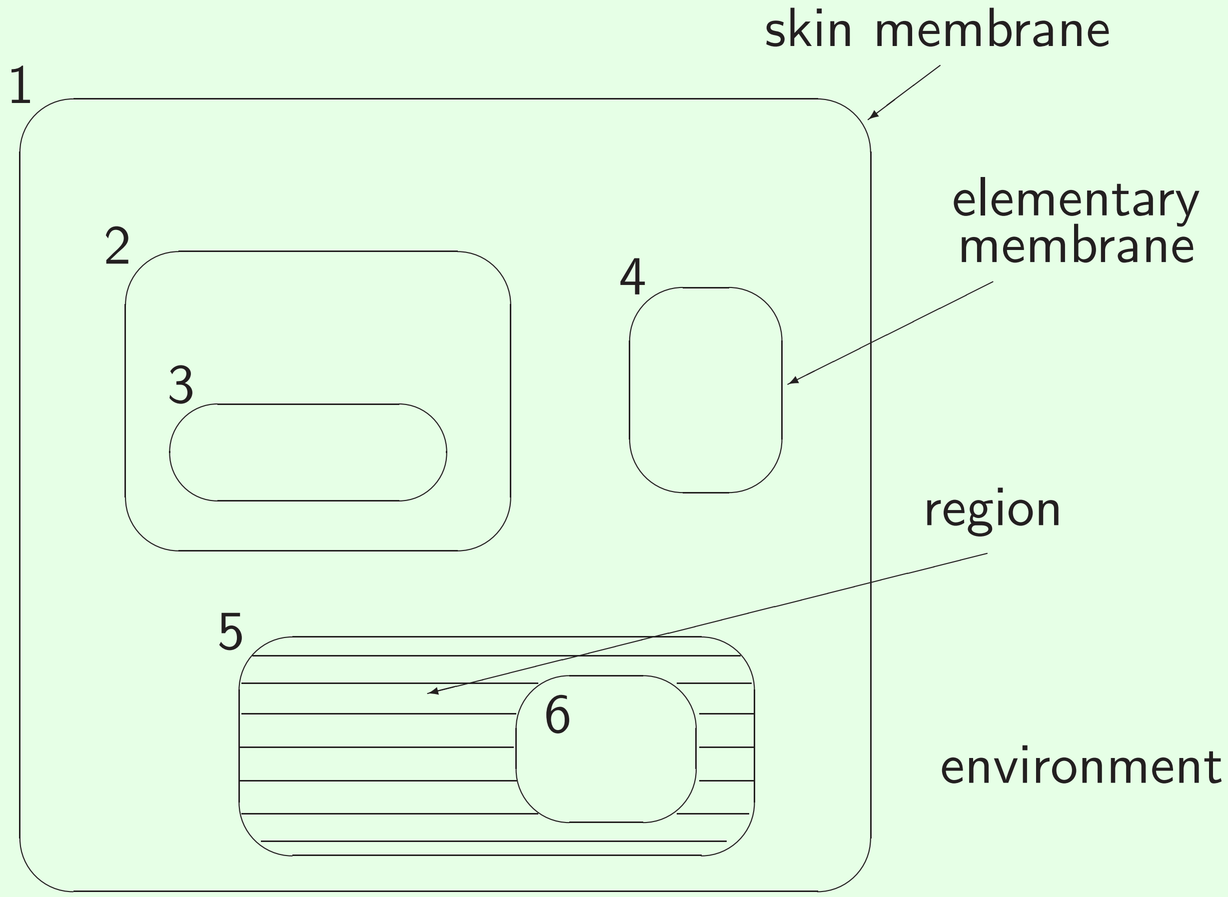
...still, not too satisfied (with DNA computing)

Let's go to the cell!

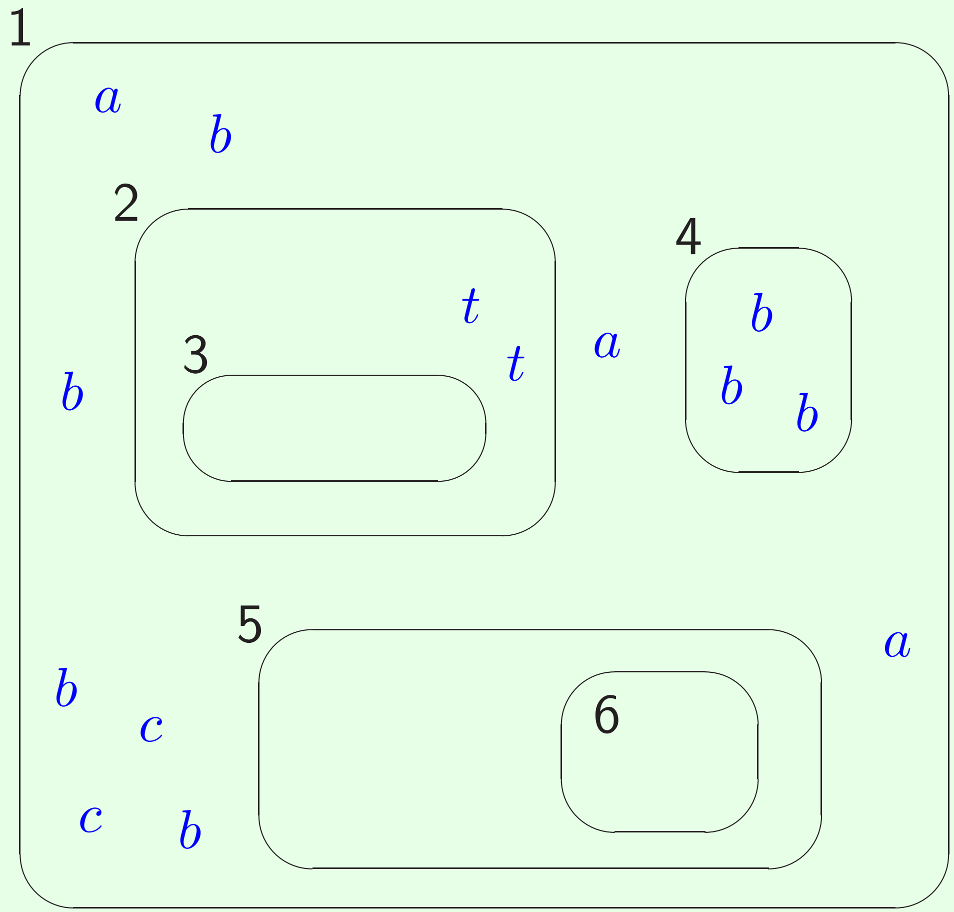


...what a jungle!

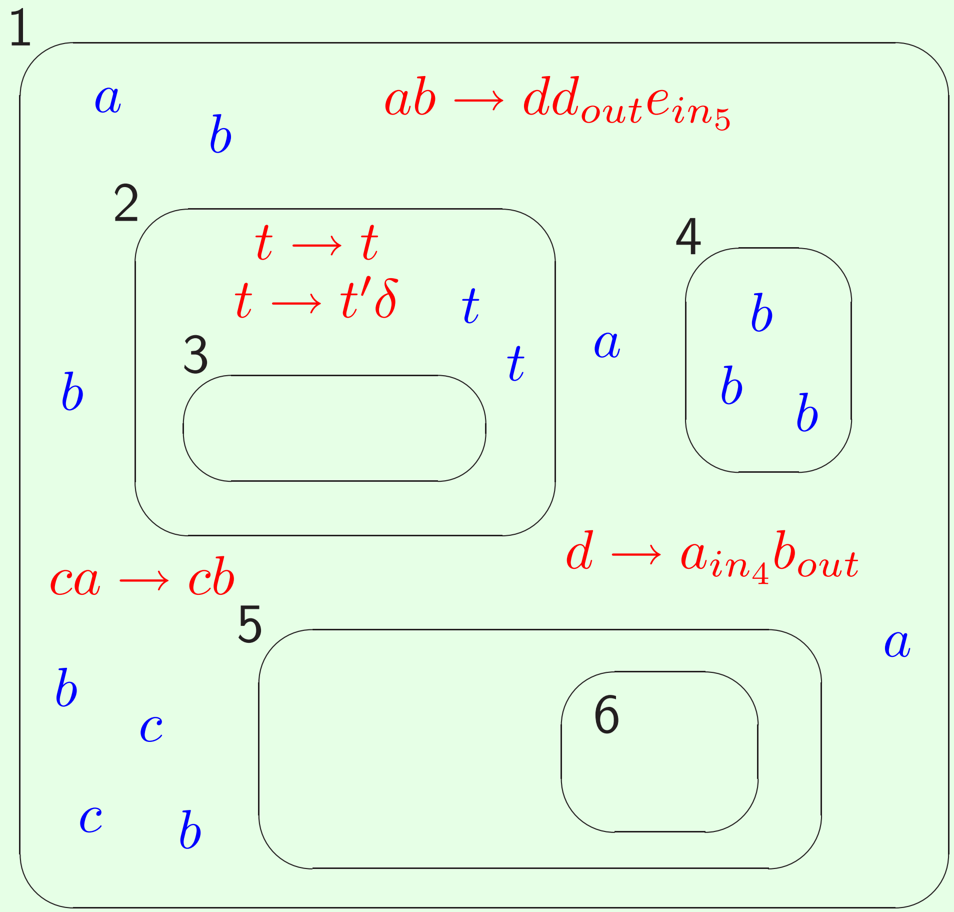
BUT, WE (THE MATHEMATICIANS) CAN SIMPLIFY:



BUT, WE (THE MATHEMATICIANS) CAN SIMPLIFY:



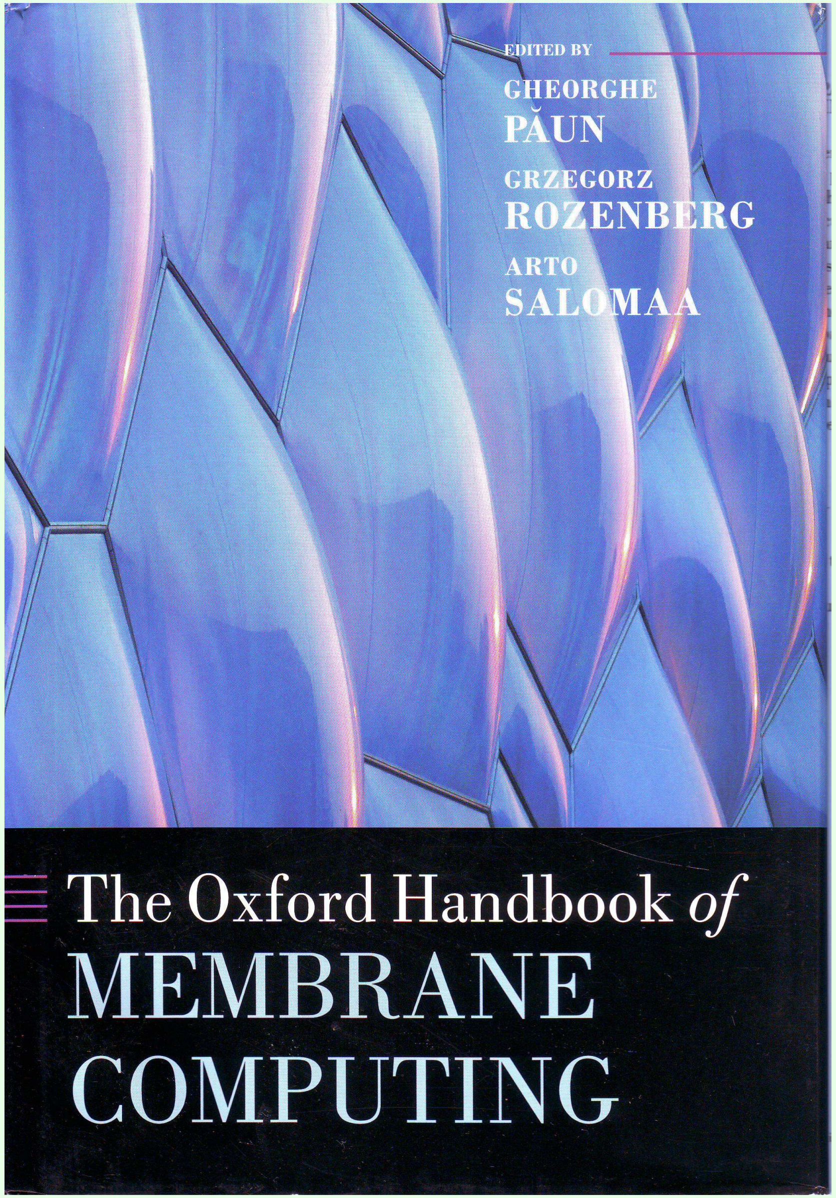
BUT, WE (THE MATHEMATICIANS) CAN SIMPLIFY:



Functioning (basic ingredients):

- nondeterministic choice of rules and objects
- maximal parallelism
- transition, computation, halting
- internal output, external output

Result: Cell-like P system


EDITED BY
GHEORGHE
PĂUN
GRZEGORZ
ROZENBERG
ARTO
SALOMAA
The Oxford Handbook of
MEMBRANE
COMPUTING

Handbook of Membrane Computing

Editors: Gheorghe Păun (Bucharest, Romania)
Grzegorz Rozenberg (Leiden, The Netherlands)
Arto Salomaa (Turku, Finland)

Advisory Board: E. Csuhaj-Varjú (Budapest, Hungary)
R. Freund (Vienna, Austria)
M. Gheorghe (Sheffield, UK)
O.H. Ibarra (Santa Barbara, USA)
V. Manca (Verona, Italy)
G. Mauri (Milan, Italy)
M.J. Pérez-Jiménez (Seville, Spain)

Oxford University Press, 2010

Introducing MC through 12 basic ideas:

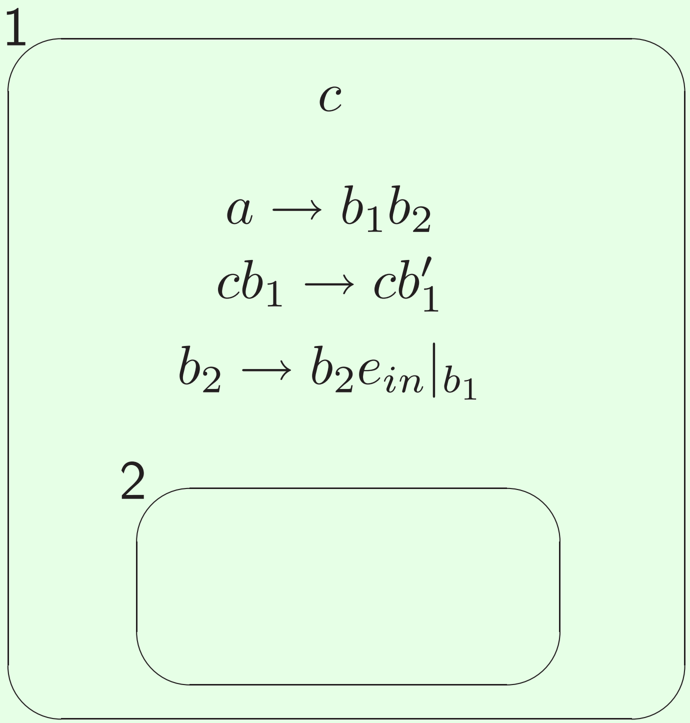
1. Cell-like P system

$$\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_o),$$

where:

- O = alphabet of objects
- μ = (labeled) membrane structure of degree m
- w_i = strings/multisets over O
- R_i = sets of evolution rules
  typical form $ab \rightarrow (a, here)(c, in_2)(c, out)$
- i_o = the output membrane

EXAMPLE



Computing system: $n \longrightarrow n^2$ (catalyst, promoter, determinism, internal output)

Input (in membrane 1): a^n

Output (in membrane 2): e^{n^2}

Computational power (Universality)

Families $NOP_m(\alpha, tar)$, $\alpha \in \{coo, ncoo, cat\} \cup \{cat_i \mid i \geq 1\}$, $m \geq 1$ or $m = *$.

Lemma 1. (collapsing hierarchy) $NOP_*(\alpha, tar) = NOP_m(\alpha, tar,)$

for all $\alpha \in \{ncoo, cat, coo\}$ and $m \geq 2$.

Theorem 1. $NOP_*(ncoo, tar) = NOP_1(ncoo) = NCF$.

Proof: use Lemma 1 and CD grammar systems

Theorem 2. $NOP_*(coo, tar) = NOP_m(coo, tar) = NRE$, for all $m \geq 1$.

Theorem 3. [Sosik: 8], [Sosik, Freund: 6], [Freund, Kari, Sosik, Oswald: 2]

$$NOP_2(cat_2, tar) = NRE$$

Conjecture $NRE - NOP_*(cat_1) \neq \emptyset$

2. String objects:

...processed by string operations:

- rewriting
- splicing (DNA computing)
- other DNA-inspired operations

More complex objects, e.g., arrays

3. Computing by communication: symport-antiport

$(ab, in), (ab, out)$ – symport (in general, $(x, in), (x, out)$)
$(a, in; b, out)$ – antiport (in general, $(u, in; v, out)$)

$(\max(|x|, |y|) =$ weight)

System

$$\Pi = (O, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m, i_o),$$

where $E \subseteq O$ is the set of objects which appear in the environment in arbitrarily many copies

Families $NOP_m(sym_p, anti_q)$

Power: (universality)

Theorem 4.

$NRE = NOP_1(sym_0, anti_2) = NOP_2(sym_2, anti_0) = NOP_1(sym_3, anti_0) = NOP_3(sym_1, anti_1)$

More general rules:

$u]_i v \rightarrow u']_i v'$ – boundary (Manca, Bernardini)
$ab \rightarrow a_{tar_1} b_{tar_2}$ – communication (Sosik)
$ab \rightarrow a_{tar_1} b_{tar_2} c_{come}$
$a \rightarrow a_{tar}$

4. Active membranes:

$a[\ ]_i \rightarrow [b]_i$	go in
$[a]_i \rightarrow b[\ ]_i$	go out
$[a]_i \rightarrow b$	membrane dissolution
$a \rightarrow [b]_i$	membrane creation
$[a]_i \rightarrow [b]_j[c]_k$	membrane division
$[a]_i[b]_j \rightarrow [c]_k$	membrane merging
$[a]_i[\ ]_j \rightarrow [[b]_i]_j$	endocytosis
$[[a]_i]_j \rightarrow [b]_i[\ ]_j$	exocytosis
$[u]_i \rightarrow [\ ]_i[u]_{@j}$	gemmation
$[Q]_i \rightarrow [O-Q]_j[Q]_k$	separation

and others

5. tissue-like P systems - membranes in the nodes of a graph
   population P systems
6. using P systems in the accepting mode
   P automata
7. trace languages
8. numerical P systems

Basic idea: numerical variables in regions, evolving by "production functions", whose value is distributed according to "repartition protocols"; dynamical systems approach (sequences of configurations), but also computing device (the set of values of a specified variable).

Example:

1

$$x_{1,1}[1]$$

$$2x_{1,1}^2 \longrightarrow 1|x_{1,1} + 1|x_{1,2}$$

2

$$x_{1,2}[3],\ x_{2,2}[1],\ x_{3,2}[0]$$

$$x_{1,2}^3 - x_{1,2} - 3x_{2,2} - 9 \longrightarrow 1|x_{2,2} + 1|x_{3,2} + 1|x_{2,3}$$

3

$$x_{1,3}[2],\ x_{2,3}[1]$$

$$2x_{1,3} - 4x_{2,3} + 4 \longrightarrow 2|x_{1,3} + 1|x_{2,3} + 1|x_{1,2}$$

4

$$x_{1,4}[2],\ x_{2,4}[2],\ x_{3,4}[2]$$

$$x_{1,4}x_{2,4}x_{3,4} \longrightarrow 1|x_{1,4} + 1|x_{2,4} + 1|x_{3,4} + 1|x_{3,2}$$

Results:

Theorem 5.

$$SLIN_1^+ \subset DSET^+P_*(poly^1(1), nneg, div)$$
$$N^+RE = SET^+P_8(poly^5(5), div) = SET^+P_7(poly^5(6), div)$$

+ many research topics and open problems

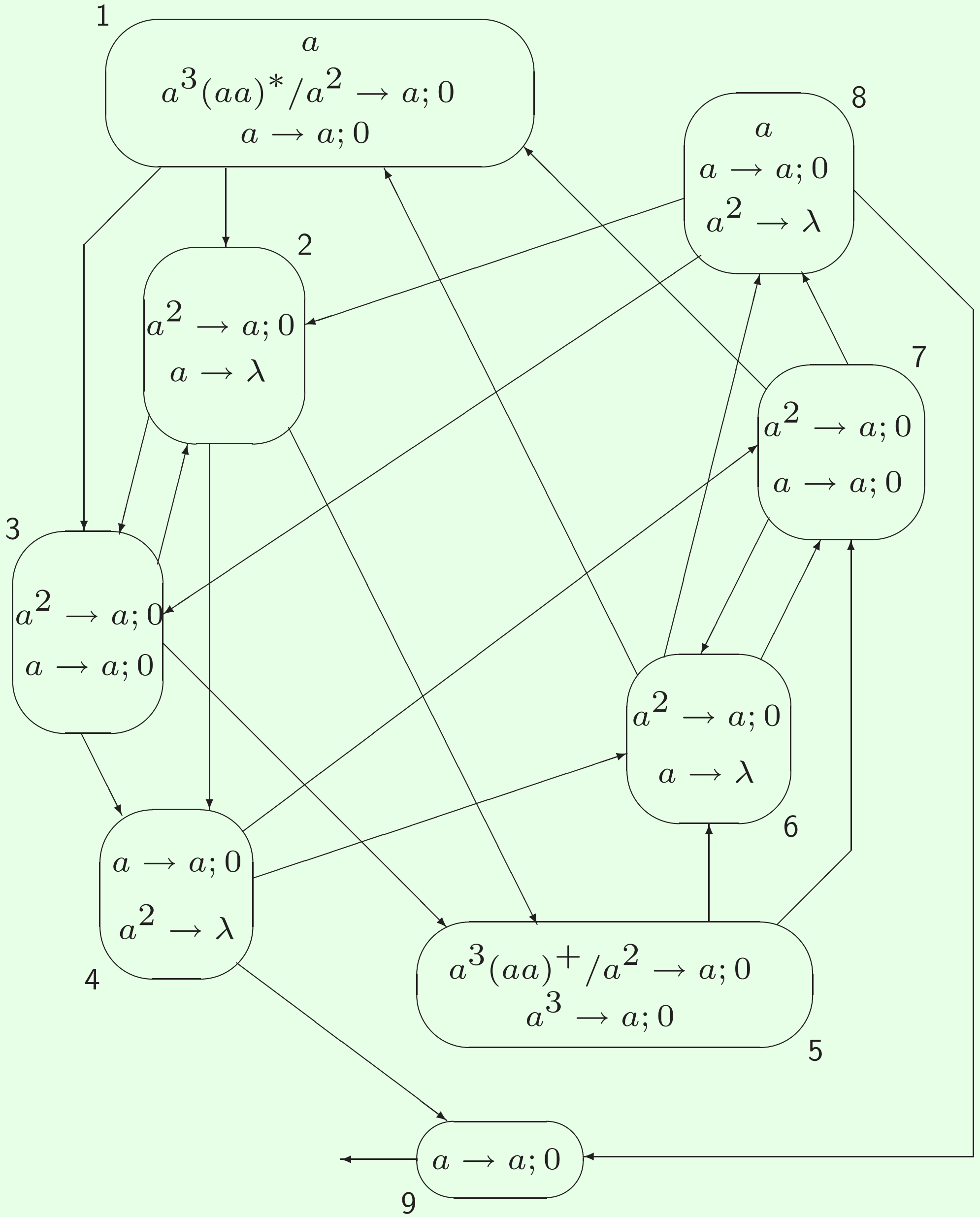
9. P systems with objects on membranes
   (brane calculi inspired P systems)

10. P colonies (set of cells of a bounded capacity, with minimal object processing rules)

11. spiking neural P systems

W. Maass movie about spiking neurons:

http://www.igi.tugraz.at/tnatschl/spike_trains_eng.html


1
a
$a^3(aa)^*/a^2 \to a; 0$
$a \to a; 0$
8
a
$a \to a; 0$
$a^2 \to \lambda$
2
$a^2 \to a; 0$
$a \to \lambda$
7
$a^2 \to a; 0$
$a \to a; 0$
3
$a^2 \to a; 0$
$a \to a; 0$
$a^2 \to a; 0$
$a \to \lambda$
6
$a \to a; 0$
$a^2 \to \lambda$
4
$a^3(aa)^+/a^2 \to a; 0$
$a^3 \to a; 0$
5
$a \to a; 0$
9

We get

$$st(\Pi) = 0^4 10^3 10^5 10^4 10^6 10^5 10^7 10^6 \ldots,$$

that is, an infinite sequence of blocks of the form $0^{2i}10^{2i-1}10^{2i+1}10^{2i}1$ with $i \geq 2$.

For $g : \{0,1\}^* \longrightarrow \{0,1\}^*$ defined by

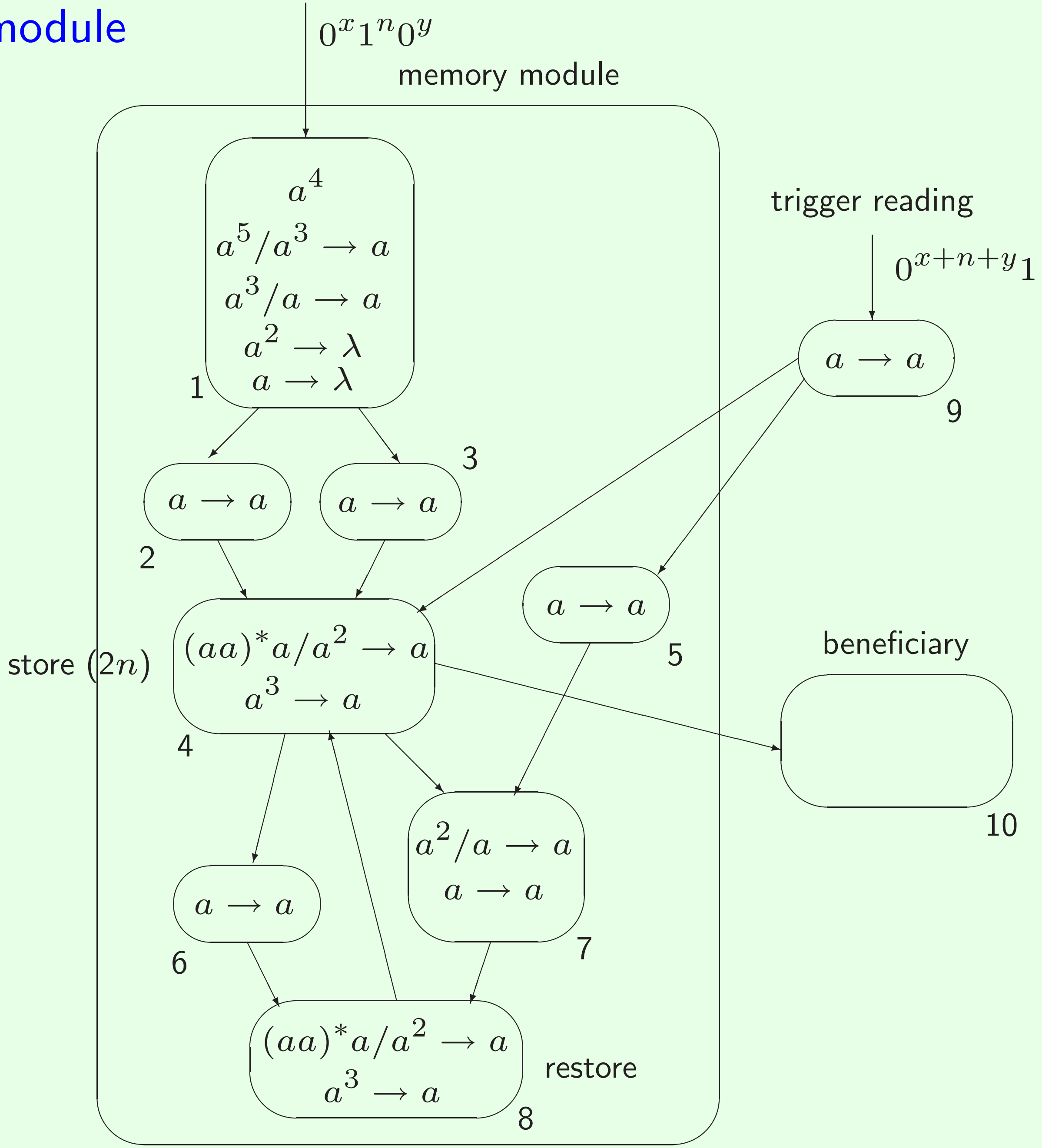
$$g(0^i 10^j 1) = 0^{i+1}10^{j+1}1,$$
$$g(w10^i 10^j 1) = 0^{i+1}10^{j+1}1,$$

for all $w \in \{0,1\}^*$ and $i, j \geq 1$, define the infinite sequence f_ω as the limit of the following sequence of strings:

$$f_0 = 0^4 10^3 1,$$
$$f_{n+1} = f_n \, g(f_n), \text{ for } n \geq 0.$$

Then $st(\Pi) = f_\omega$.

An SN memory module



Formal definition:

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out),$$

where:

1. $O = \{a\}$ is the singleton alphabet (a is called *spike*);

2. $\sigma_1, \ldots, \sigma_m$ are *neurons*, of the form

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m,$$

where:

a) $n_i \geq 0$ is the *initial number of spikes* contained by the neuron;
b) R_i is a finite set of *rules* of the following two forms:

(1) $E/a^c \to a; d$, where E is a regular expression with a the only symbol used, $c \geq 1$, and $d \geq 0$;
(2) $a^s \to \lambda$, for some $s \geq 1$, with the restriction that $a^s \in L(E)$ for no rule $E/a^c \to a; d$ of type (1) from R_i;

3. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$ ($synapses$ among neurons);

4. $in, out \in \{1, 2, \ldots, m\}$ indicate the $input$ and the $output$ $neuron$.

only out = generative system

only in = accepting system

both in, out = computing system

Spike trains, types of output

FAMILIES: $Spik_{gen}P_m(rule_k, cons_p, forg_q)$ – generative

$Spik_{acc}P_m(rule_k, cons_p, forg_q)$ – accepting ($DSpik$, if deterministic)

Theorem 6. $NFIN = Spik_{gen}P_1(rule_*, cons_1, forg_0) = Spik_{gen}P_2(rule_*, cons_*, forg_*)$.

Theorem 7. $Spik_{gen}P_*(rule_2, cons_3, forg_3) = Spik_{acc}P_*(rule_2, cons_3, forg_2) = NRE$.

Theorem 8. $SLIN_1 = Spik_{gen}P_*(rule_k, cons_p, forg_q, bound_s)$, for all $k \geq 3$, $q \geq 3$, $p \geq 3$, and $s \geq 3$.

Normal forms, generating languages and infinite sequences, small universal SN P systems, etc.

Extension (spiking requesting rules: $E/\lambda \leftarrow a^r$)

Some results (extended):

Lemma 1. *The number of configurations reachable after n steps by an extended SN P system with request rules of degree m is bounded by a polynomial $g(n)$ of degree m.*

Theorem 1. *If $f: V^+ \longrightarrow V^+$ is an injective function, $card(V) \geq 2$, then there is no extended SN P system Π with request rules such that $L_f(V) = \{x\, f(x) \mid x \in V^+\} = L_*^g(\Pi)$.*

Corollary 1. *The following two languages are not in $L_*^g SNP_*$ (in all cases, $card(V) = k \geq 2$):*

$$L_1 = \{x\, mi(x) \mid x \in V^+\}, \qquad L_2 = \{xx \mid x \in V^+\}.$$

12: dP systems

A *dP scheme* (of degree $n \geq 1$) is a construct

$$\Delta = (O, \Pi_1, \ldots, \Pi_n, R),$$

where:

1. O is an alphabet of objects;

2. $\Pi_1, \ldots, \Pi_n$ are cell-like P systems with O as the alphabet of objects and the skin membranes labeled with $s_1, \ldots, s_n$, respectively;

3. R is a finite set of rules of the form $(s_i, u/v, s_j)$, where $1 \leq i, j \leq n$, $i \neq j$, and $u, v \in O^*$, with $uv \neq \lambda$; $|uv|$ is called the *weight* of the rule $(s_i, u/v, s_j)$.

A *dP automaton* is a construct

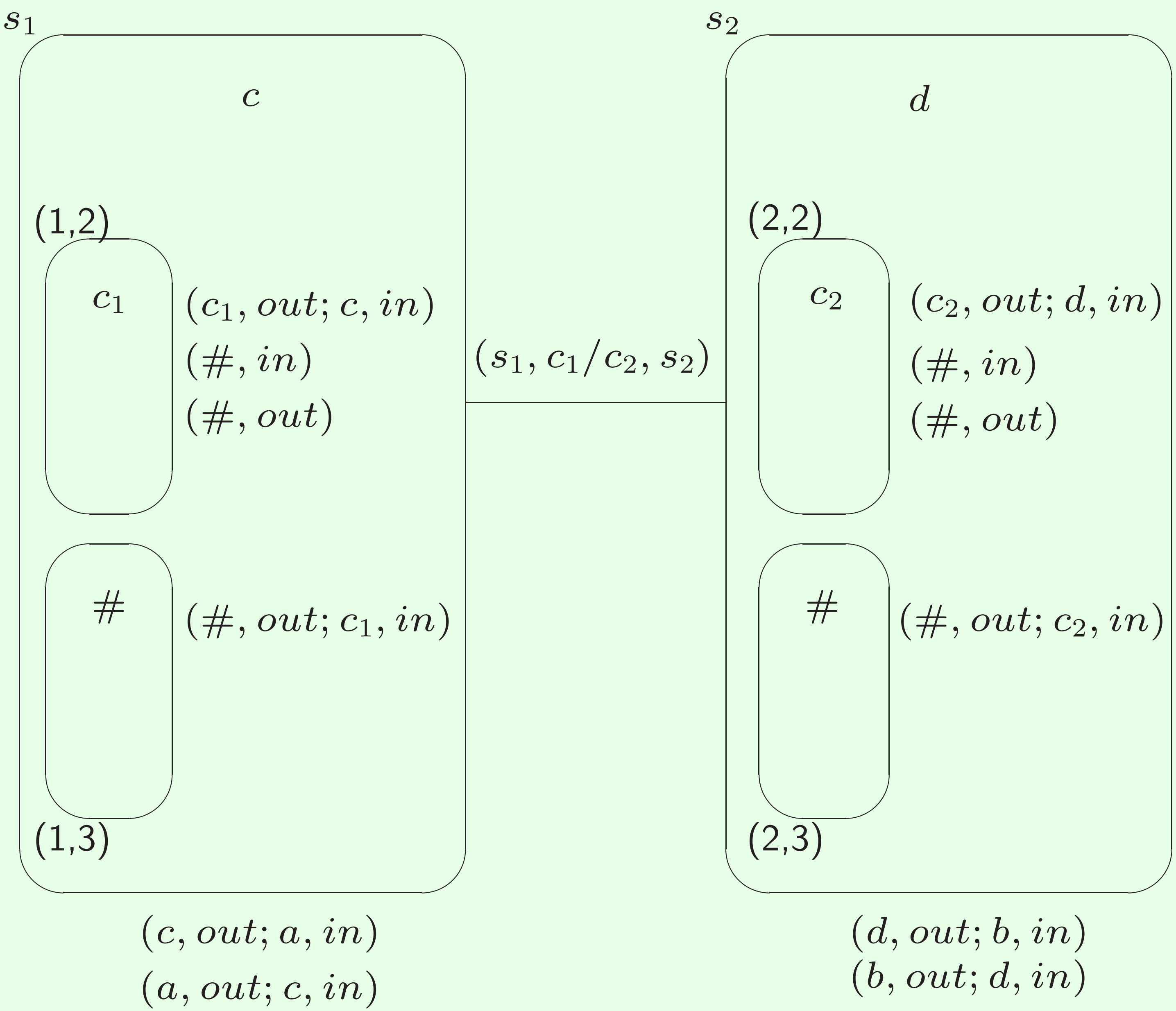
$$\Delta = (O, E, \Pi_1, \ldots, \Pi_n, R),$$

where $(O, \Pi_1, \ldots, \Pi_n, R)$ is a dP scheme, $E \subseteq O$ (the objects available in arbitrarily many copies in the environment), $\Pi_i = (O, \mu_i, w_{i,1}, \ldots, w_{i,k_i}, E, R_{i,1}, \ldots, R_{i,k_i})$ is a symport/antiport P system of degree k_i (without an output membrane), with the skin membrane labeled with $(i, 1) = s_i$, for all $i = 1, 2, \ldots, n$.

A halting computation with respect to Δ accepts the string $x = x_1 x_2 \ldots x_n$ over O if the components $\Pi_1, \ldots, \Pi_n$, starting from their initial configurations, using the symport/antiport rules as well as the inter-components communication rules, in the non-deterministically maximally parallel way, bring from the environment the substrings $x_1, \ldots, x_n$, respectively, and eventually halts.

Communication complexity, power, [efficiently] parallelizable languages, etc.

A dP system accepting the language $L(\Delta) = \{(ac)^s(bd)^s \mid s \geq 0\}$.



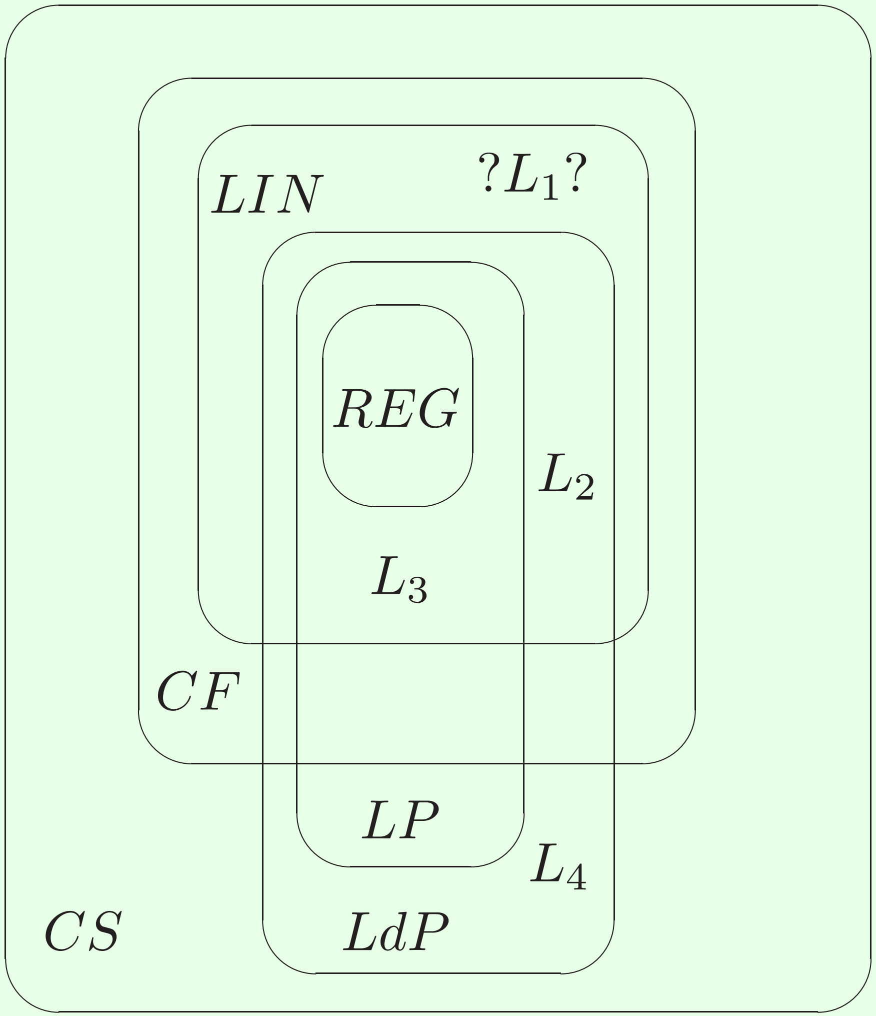


Figure 1: The place of the families LP and LdP in Chomsky hierarchy

SN dP Systems

An $SN\ dP\ system$ is a construct

$$\Delta = (O, \Pi_1, \ldots, \Pi_n, esyn), \text{ where:}$$

1. $O = \{a\}$ ($a =$ spike),

2. $\Pi_i = (O, \sigma_{i,1}, \ldots, \sigma_{i,k_i}, syn, in_i)$ is an SN P system with request rules present only in neuron σ_{in_i} – problem: relax this ($\sigma_{i,j} = (n_{i,j}, R_{i,j})$),

3. $esyn$ is a set of *external synapses*, namely between neurons from different systems Π_i, with the restriction that between two systems Π_i, Π_j there exist at most one link from a neuron of Π_i to a neuron of Π_j and at most one link from a neuron of Π_j to a neuron of Π_i.

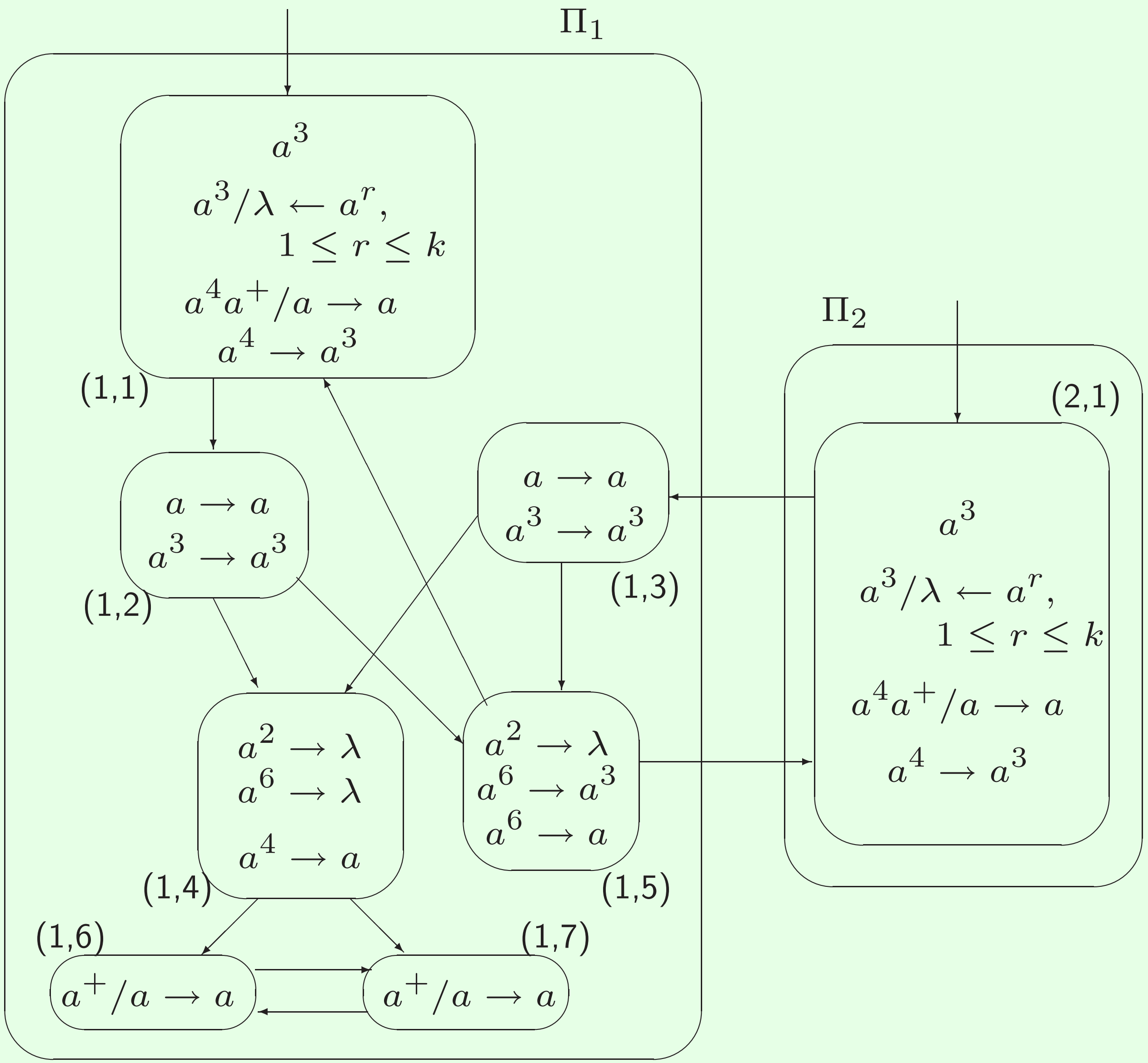
The systems $\Pi_i, 1 \leq i \leq n$, are called *components* (or *modules*) of the system Δ.

Languages, families:

$L(\Delta)$ is the set of all strings $x \in V^*$ such that we can write $x = x_1x_2 \ldots x_n$, with $||x_i| - |x_j|| \leq 1$ for all $1 \leq i, j \leq n$, each component Π_i of Δ takes as input the string $x_i, 1 \leq i \leq n$, and the computation halts. Moreover, we can distinguish between considering b_0 as a symbol or not, like above, thus obtaining the languages $L_\alpha(\Delta)$, with $\alpha \in \{0, 1, 2, \ldots\} \cup \{\infty, *\}$.

$L_\alpha SNdP_n$, the family of languages $L_\alpha(\Delta)$, for Δ of degree at most n and $\alpha \in \{0, 1, 2, \ldots\} \cup \{\infty, *\}$.

Example $(\{ww \mid w \in \{b_1, b_2, \ldots, b_k\}^*\} \in L_{k+2}SNdP_2)$



Results (in general):

- characterization of Turing computability (RE, NRE, $PsRE$)
  Examples: by catalytic P systems (2 catalysts) [Sosik, Freund, Kari, Oswald]
  by (small) symport/antiport P systems [many]

- polynomial solutions to NP-complete problems – even characterizations of PSPACE (by using an exponential workspace
  created in a "biological way": membrane division, membrane creation, string replication, etc) [Sevilla team], [Milano team], [Obtulowicz], [Alhazov, Pan], [Madrid team] etc

- other types of mathematical results (normal forms, hierarchies, determinism versus nondeterminism, complexity) [Ibarra group]

- connections with ambient calculus, Petri nets, X-machines, quantum computing, lambda calculus, brane calculus, etc. [many]

- simulations and implementations (Adelaide, Sevilla, Madrid)

- applications

The most practical application

Open problems, research topics:

Many: see the P page

- borderlines: universality/non-universality, efficiency/non-efficiency (local problems: the power of 1 catalyst, the role of polarizations, dissolution, etc. general problems: uniform versus semi-uniform, deterministic-confluent, pre-computed resources, etc.)
- semantics (events, causality, etc.)
- neural-like systems (more biology, complexity, applications, etc.)
- user friendly, flexible, efficient (!) software for bio-applications
- MC and economics
- implementations (electronics, bio-lab), dedicated hardware and software (P-lingua)
- finding a killer-app

Applications:

- biology, medicine, ecosystems (continuous versus discrete mathematics) [Sevilla, Verona, Milano, Sheffield, Nottingham, Ruston, etc.]

- computer science (computer graphics, sorting/ranking, 2D languages, cryptography, general model of distributed-parallel computing) [many]

- linguistics (modeling framework, parsing) [Tarragona, Chișinău]

- optimization (membrane algorithms [Nishida, 2004], [many - especially in China])

- economics ([Warsaw group], [R. Păun], [Vienna group])

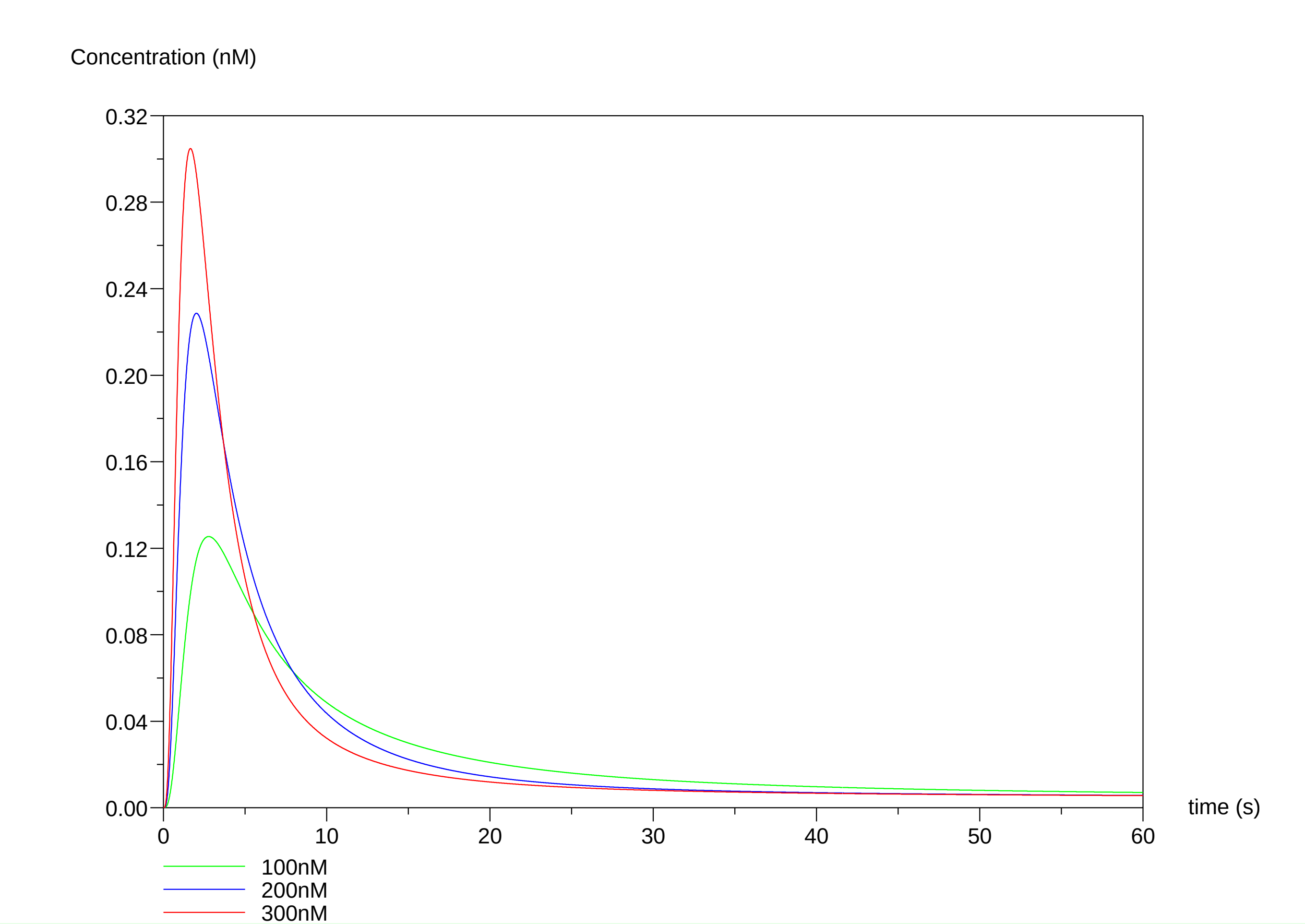
Applications of MC in biology, bio-medicine, ecology – several chapters in *Handbook*

A typical application in biology/medicine:

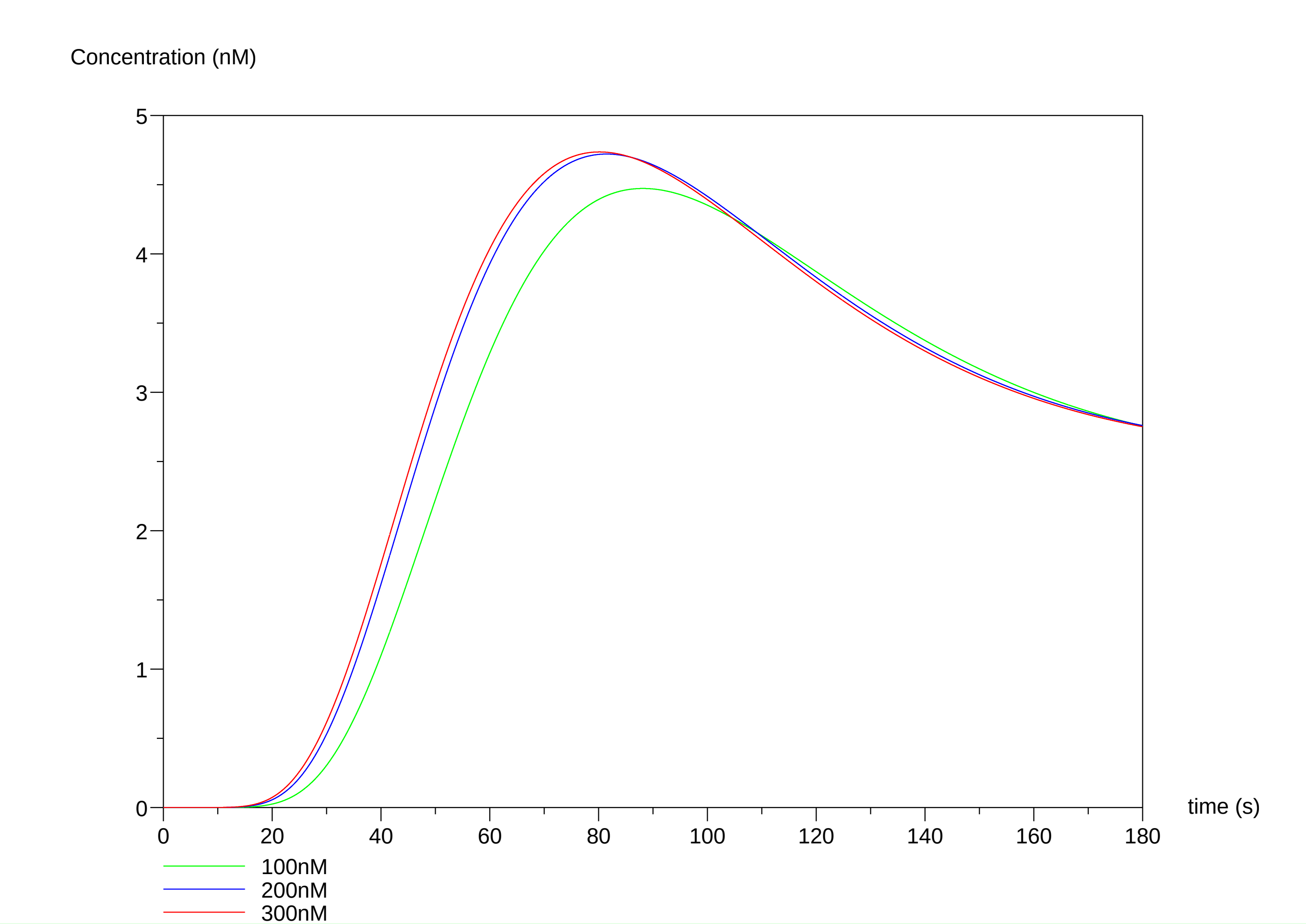
M.J. Pérez–Jiménez, F.J. Romero–Campero:
A Study of the Robustness of the EGFR Signalling Cascade Using Continuous Membrane Systems.
In *Mechanisms, Symbols, and Models Underlying Cognition. First International Work-Conference on the Interplay between Natural and Artificial Computation*, IWINAC 2005 (J. Mira, J.R. Alvarez, eds.), LNCS 3561, Springer, Berlin, 2005, 268–278.

- 60 proteins, 160 reactions/rules
- reaction rates from literature
- results as in experiments

Typical outputs:



The EGF receptor activation by auto-phosphorylation
(with a rapid decay after a high peak in the first 5 seconds)



The evolution of the kinase MEK
(proving a surprising robustness of the signalling cascade)

Other bio-applications:

- photosynthesis [Nishida, 2002]
- Brusselator [Suzuki, Verona, Milano]
- quorum sensing in bacteria [Nottingham, Sheffield, Sevilla]
- cancer related pathways [Sevilla, Ruston-Louisiana]
- circadian cycles [Verona]
- apoptosis [Ruston-Lousiana]
- signaling pathways in yeast [Milano]
- HIV infection [Edinburgh, Ruston-Louisiana]
- peripheral proteins [Trento]
- others [Milano, Iași, Bucharest, Sevilla, Verona, etc.]

Modeling ecosystems

Y. Suzuki, H. Tanaka, Artificial life and P systems, WMC1, Curtea de Argeș, 2000 (herbivorous, carnivorous, volatiles)

Lotka-Voltera model (predator-prey) [Verona, Milano]

M. Cardona, M.A. Colomer, M.J. Perez-Jimenez, S. Danuy, A. Margalida, A P system modeling an ecosystem related to the bearded vulture, 6BWMC

(Some) Results:

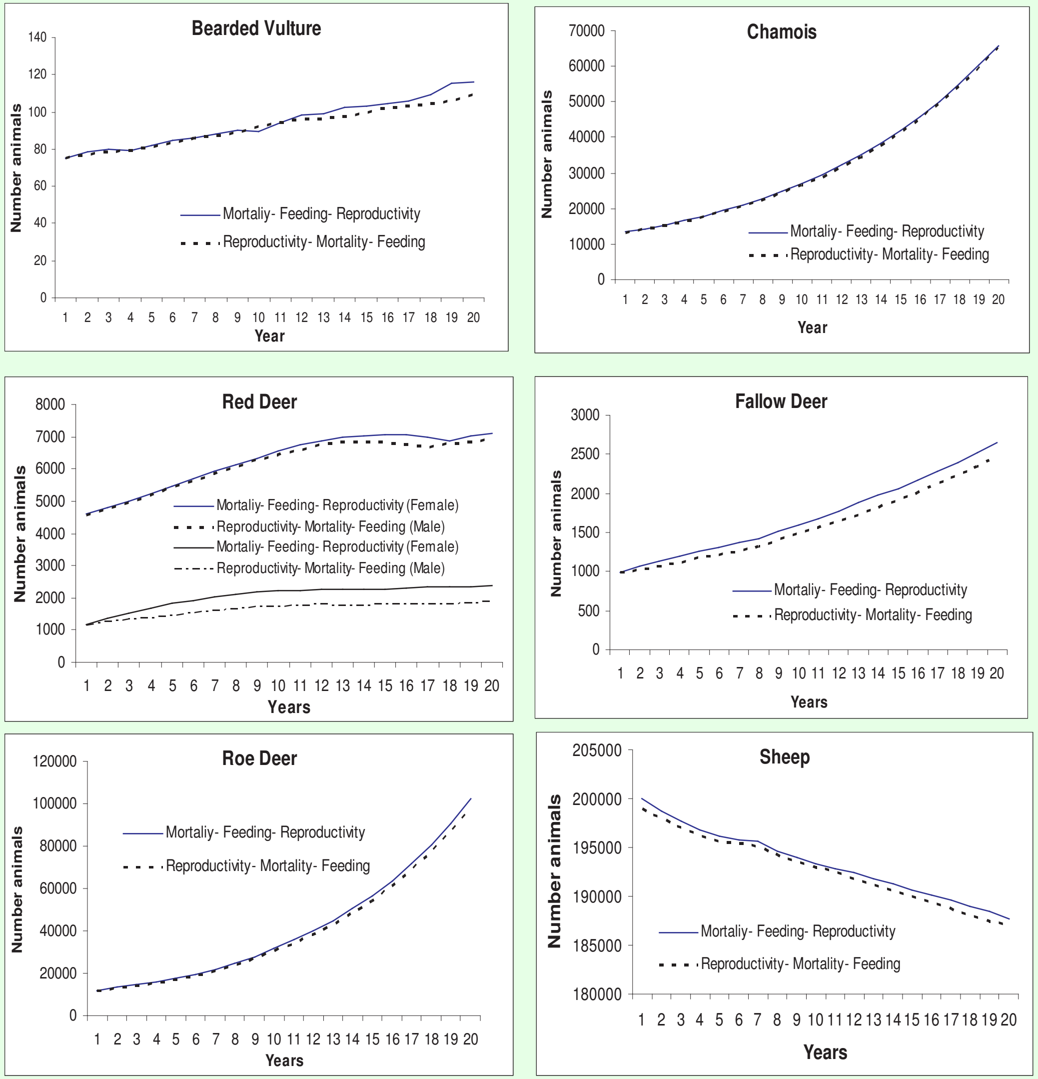


Figure 2: Robustness of the ecosystem

Membrane algorithms – *T. Nishida*

- candidate solutions in regions, processed locally (local sub-algorithms)
- better solutions go down
- static membrane structure – dynamical membrane structure
- two-phases algorithms

Excellent solutions for Travelling Salesman Problem (benchmark instances)

- rapid convergence
- good average and worst solutions (hence reliable method)
- in most cases, better solutions than simulated annealing

Still, many problems remains: check for other problems, compare with sub-algorithms, more membrane computing features, parallel implementations (no free lunch theorem)

Recent: L. Huang, N. Wang, J. Tao; G. Ciobanu, D. Zaharie; A. Leporati, D. Pagani; M. Gheorghe et colab.

SOFTWARE AND APPLICATIONS:

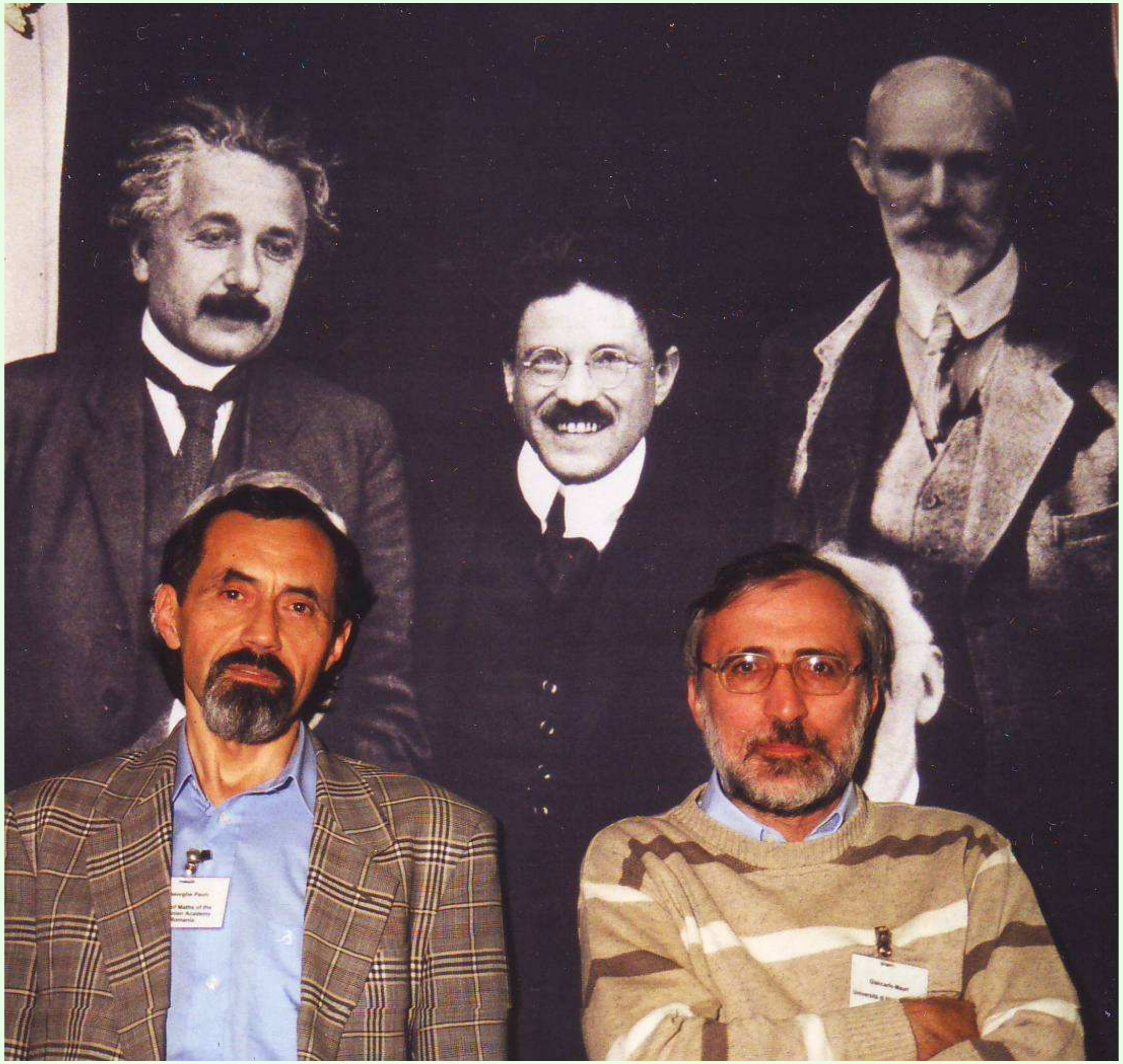
Verona (Vincenzo Manca: `vincenzo.manca@univr.it`)

Sheffield (Marian Gheorghe: `M.Gheorghe@dcs.shef.ac.uk`)

Sevilla (Mario Pérez-Jiménez: `marper@us.es`) – P-lingua!

Milano (Giancarlo Mauri: `mauri@disco.unimib.it`)

Trento, Nottingham, Leiden/Edinburgh, Vienna, Evry, Iași



Finally, satisfied...

Hypercomputation = passing beyond the Turing barrier

Fypercomputation = passing polynomially beyond the **NP** barrier

So far: membrane division, membrane creation, string replication, pre-computed resources

Further ideas:

- (local) acceleration (membranes, rules, objects)
- oracles
- ω multiplicity (like in R systems) – SAT solved in poly time
- what else?

Thank you!

...and please do not forget: `http://ppage.psystems.eu`

(with mirrors in China: `http://bmc.hust.edu.cn/psystems`,
`http://bmchust.3322.org/psystems`)