An Overview of Membrane Computing

Krishna Shankara Narayanan

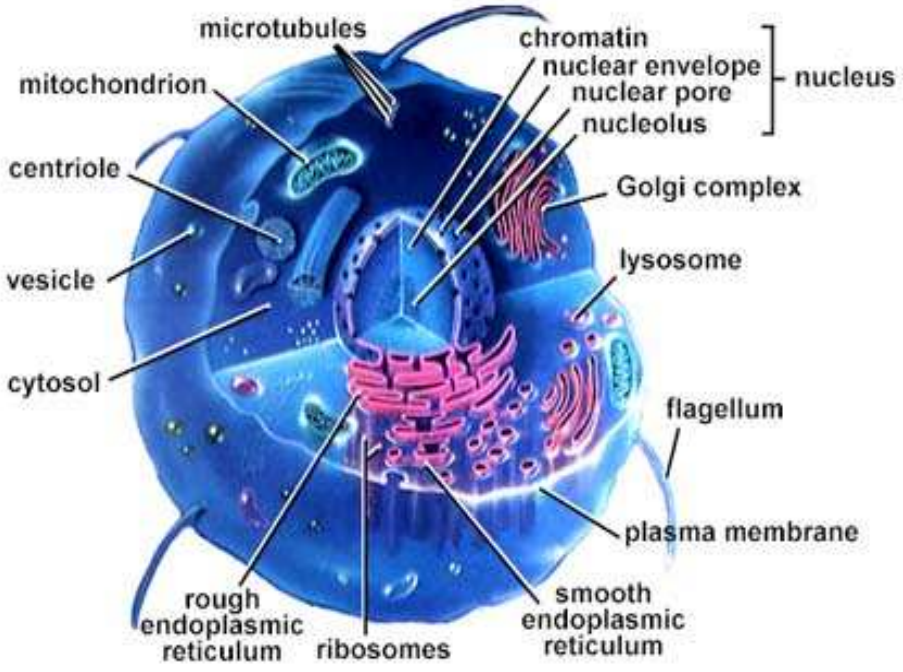
Department of Computer Science & Engineering
Indian Institute of Technology Bombay

Membrane Computing

"The paradigmatic idea of membrane computing is to see whether we can mimic the living cell - its structure and functioning"

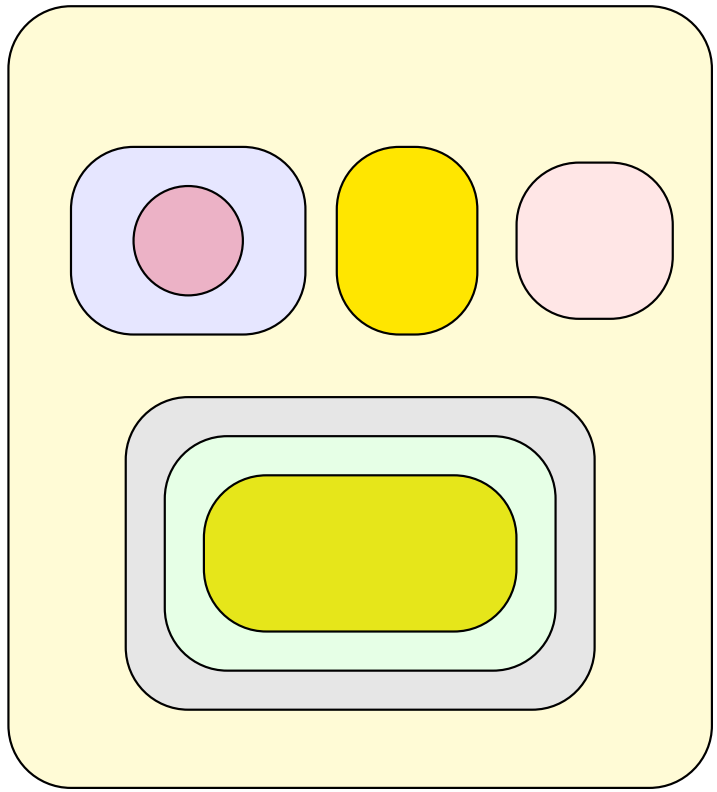
Gheorghe Păun

The Living Cell

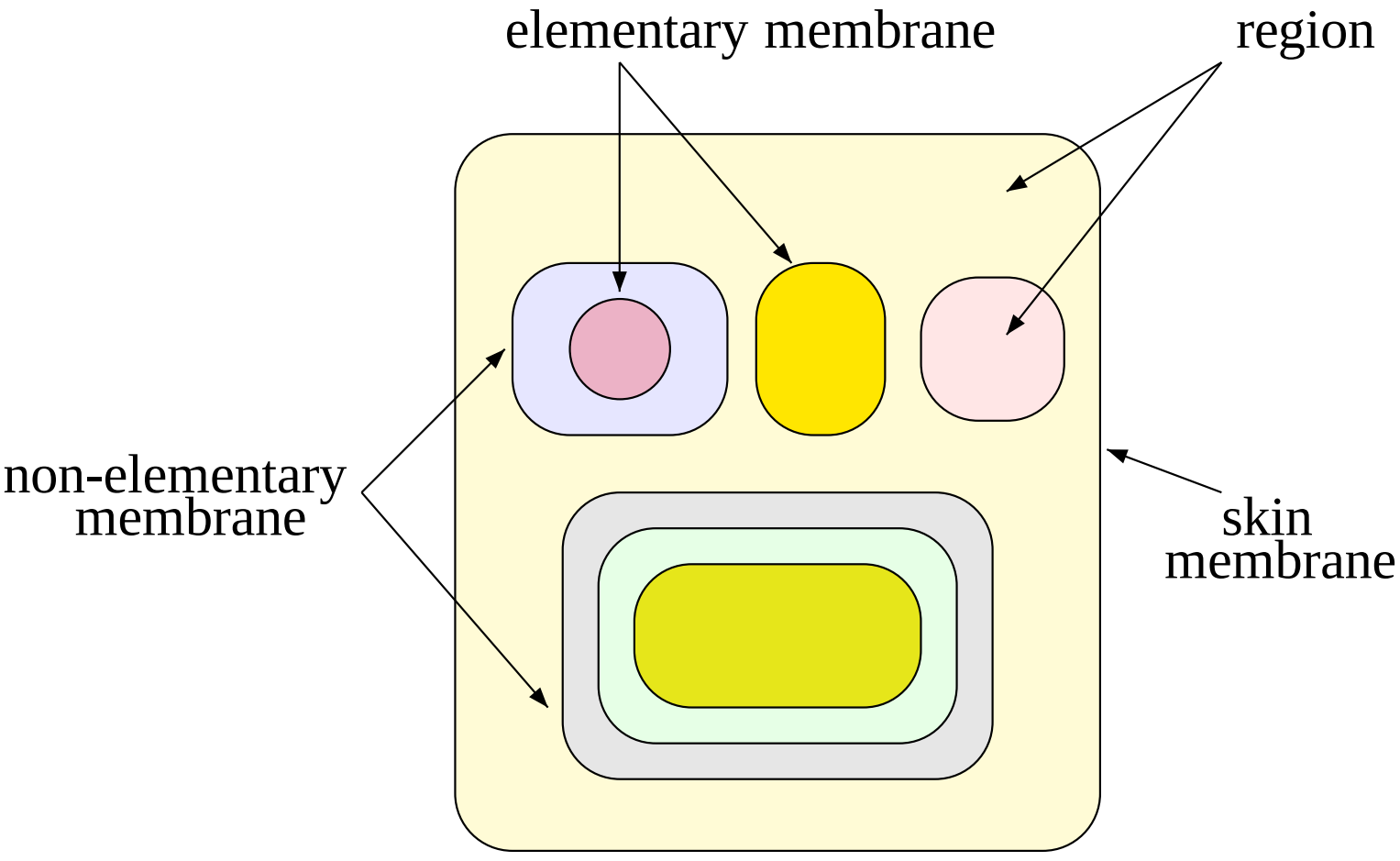


What a jungle!

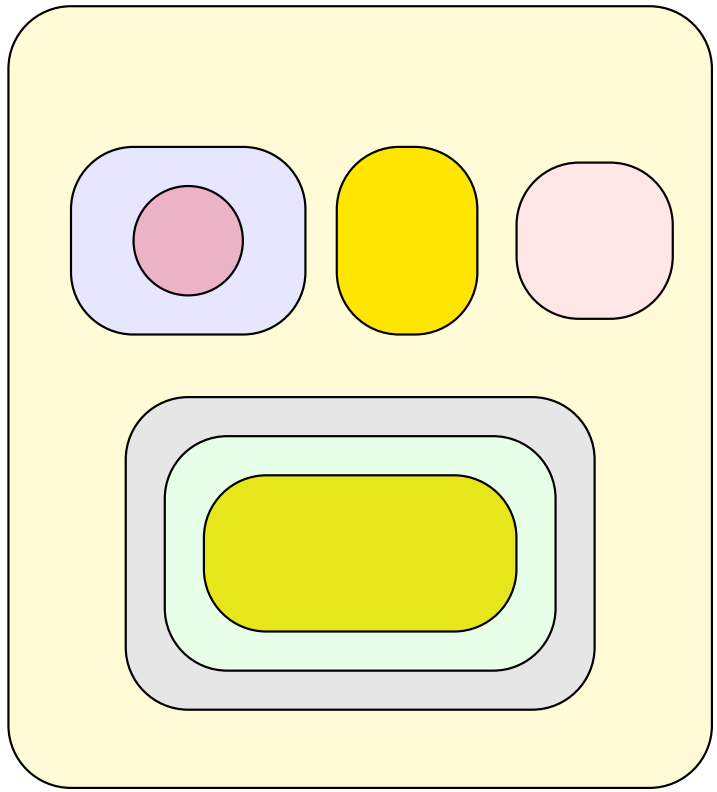
Abstraction of the Living Cell



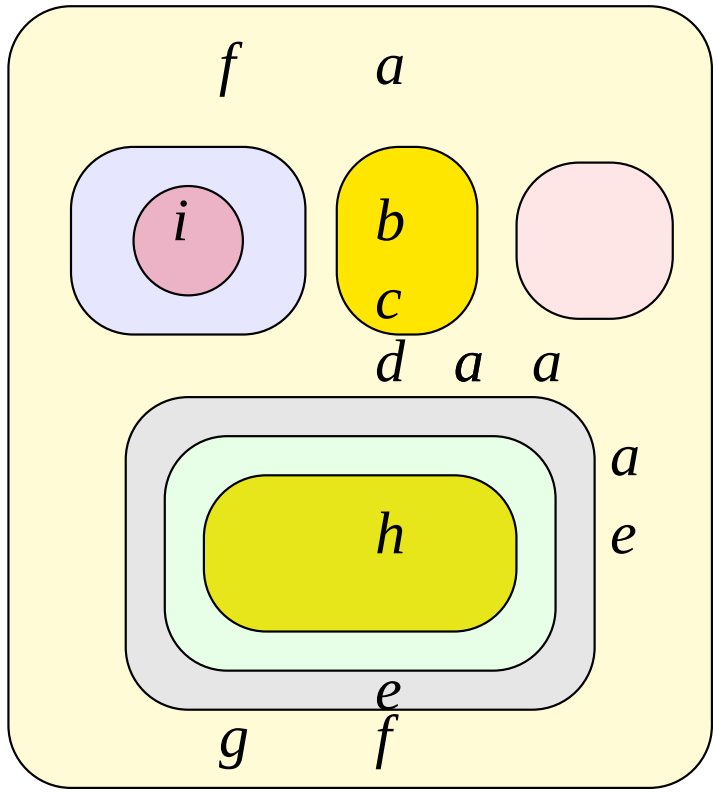
Abstraction of the Living Cell



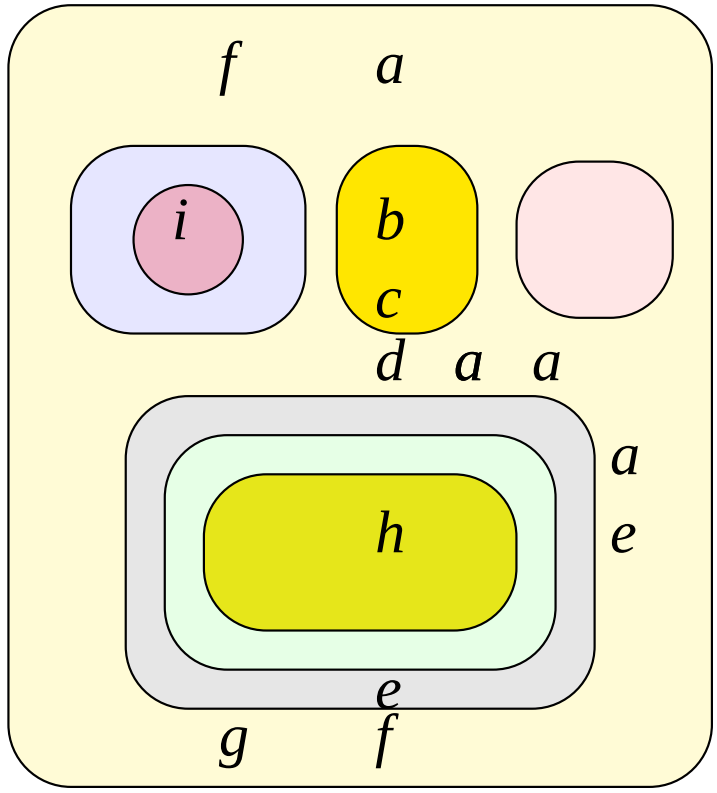
Abstraction of the Living Cell



Abstraction of the Living Cell



Abstraction of the Living Cell



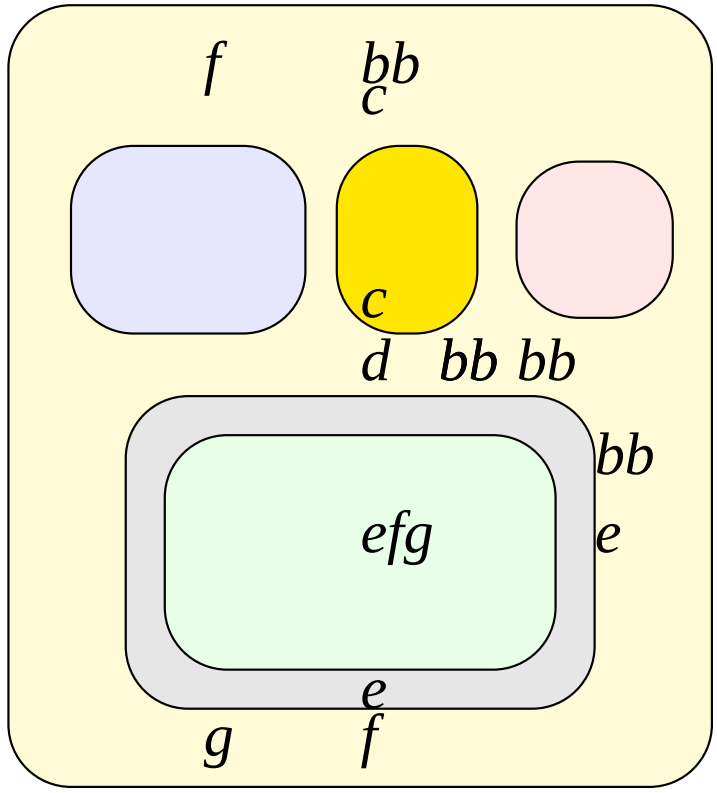
$a \rightarrow bb$

$b \rightarrow (c, out)$

$i \rightarrow \delta$

$h \rightarrow efg\delta$

Abstraction of the Living Cell



$a \rightarrow bb$

$b \rightarrow (c, out)$

$i \rightarrow \delta$

$h \rightarrow efg\delta$

Basic Ingredients

- non-deterministic choice of rules, objects
- maximal parallelism
- transitions, computation, halting
- internal output, external output

Result : Cell-like abstract system : P System

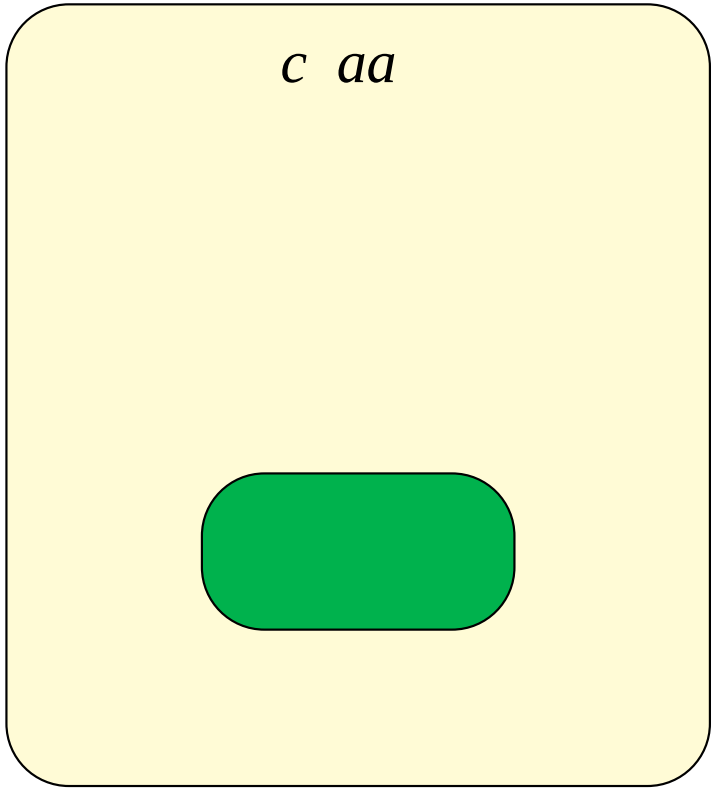
Basic Ideas in Membrane Computing

Symbol Objects (as in the example)

- $\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0)$
- μ : membrane structure
- w_i : multisets of objects in membrane i
- R_i : rules of the form $a \to (u, in)(v, out)w$ or $ca \to c(u, in)(v, out)w$, c : catalyst
- i_0 : output membrane

Example : Symbol Objects

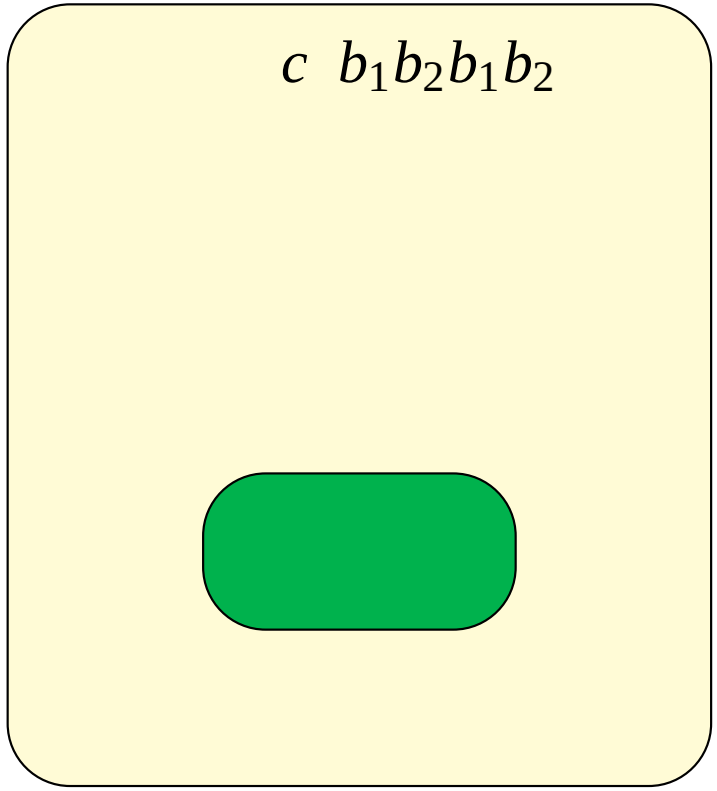
Using catalysts, promoters. Input : a^n; Output : e^{n^2}



$$a \longrightarrow b_1 b_2$$
$$cb_1 \longrightarrow cb_1'$$
$$b_2 \longrightarrow b_2(e, in)/b_1$$

Example : Symbol Objects

Using catalysts, promoters. Input : a^n; Output : e^{n^2}



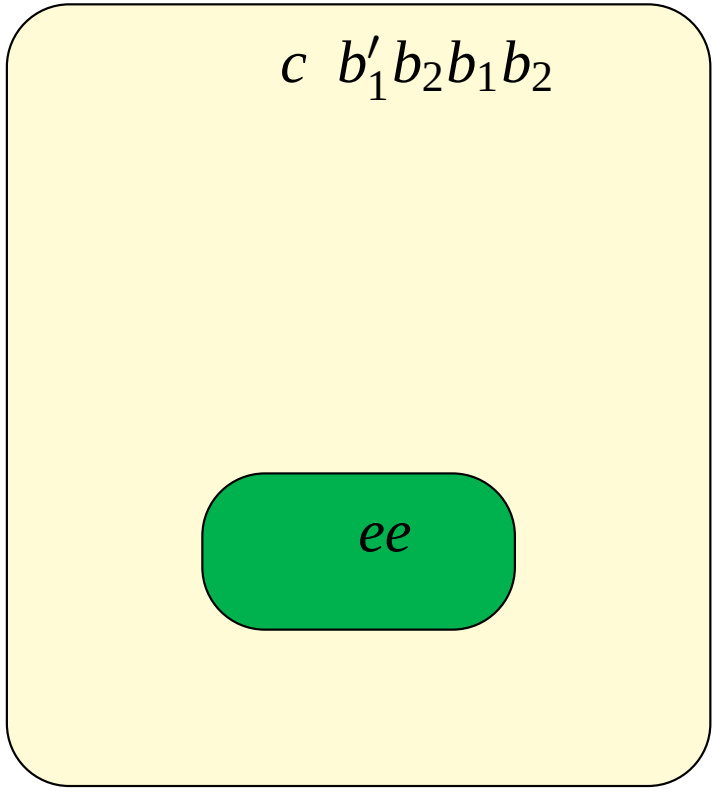
$a \rightarrow b_1 b_2$

$cb_1 \rightarrow cb_1'$

$b_2 \rightarrow b_2(e, in)/b_1$

Example : Symbol Objects

Using catalysts, promoters. Input : a^n; Output : e^{n^2}



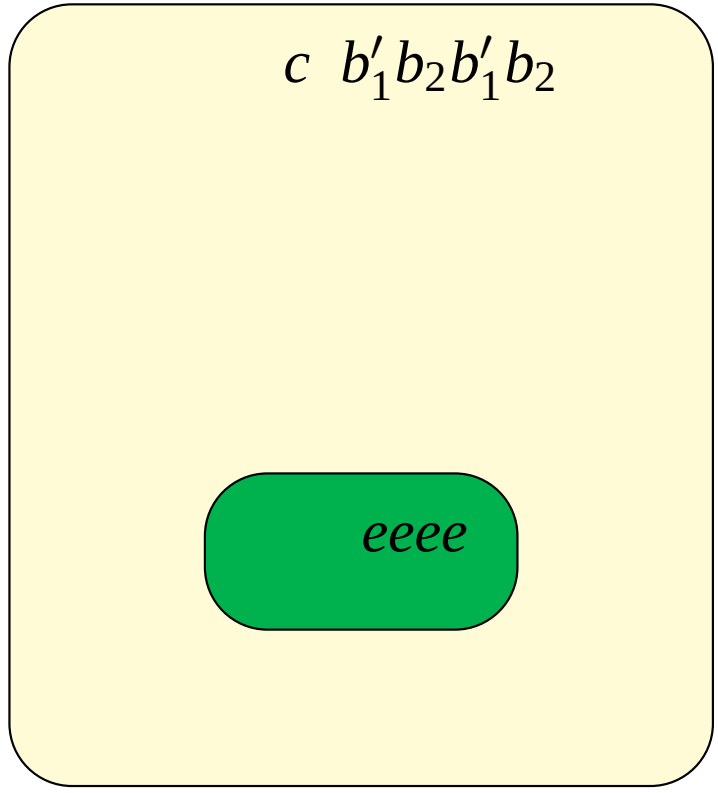
$a \rightarrow b_1 b_2$

$cb_1 \rightarrow cb_1'$

$b_2 \rightarrow b_2(e, in)/b_1$

Example : Symbol Objects

Using catalysts, promoters. Input : a^n; Output : e^{n^2}



$a \rightarrow b_1 b_2$

$cb_1 \rightarrow cb_1'$

$b_2 \rightarrow b_2(e, in)/b_1$

Computational Power

Families $\mathbb{N}OP_m(\alpha)$, $\alpha \in \{coo, ncoo, cat\} \cup \{cat_i \mid i \geq 1\}$, $m \geq 1$ or $m = *$

Lemma

(collapsing hierarchy) $\mathbb{N}OP_*(\alpha) = \mathbb{N}OP_m(\alpha)$, $\alpha \in \{coo, ncoo, cat\}$ *and* $m \geq 2$.

Theorem

$\mathbb{N}OP_*(ncoo) = \mathbb{N}OP_1(ncoo) = \mathbb{N}CF$

Theorem

$\mathbb{N}OP_2(cat_2) = \mathbb{N}RE$

Conjecture: $\mathbb{N}RE - \mathbb{N}OP_*(cat_1) \neq \emptyset$

String Objects

- rewriting rules
- inspired by splicing and (other) DNA operations
- More complex objects : arrays, pictures

Computing by communication : symport/antiport

- $(x, in); (x, out)$: symport; $(x, in; y, out)$: antiport
- $max(|x|, |y|)$: weight
- System : $\Pi = (O, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m, i_0)$
- $E \subseteq O$ is the set of objects that appear in the environment in arbitrarily many copies
- Families $\mathbb{N}OP_m(symp_p, anti_q)$

Theorem

$\mathbb{N}RE = \mathbb{N}OP_1(sym_0, anti_2) = \mathbb{N}OP_2(sym_2, anti_0) =$
$\mathbb{N}OP_1(sym_3, anti_0) = \mathbb{N}OP_3(sym_1, anti_1)$

Active Membranes

$a[\,]_i \rightarrow [b]_i$	go in
$[a]_i \rightarrow b[\,]_i$	go out
$[a]_i \rightarrow b$	dissolution
$a \rightarrow [b]_i$	membrane creation
$[a]_i \rightarrow [b]_j[c]_k$	membrane division
$[b]_j[c]_k \rightarrow [bc]_l$	membrane merging
$[u]_i \rightarrow [\,]_i[u]_j$	gemmation
$[Q]_i \rightarrow [O-Q]_j[Q]_k$	separation

Applications: Efficient solutions of intractable problems
(illustration coming up later)

Some More Variants

- Tissue-like systems : membranes in the nodes of a graph
- P automata
- Systems with objects on membranes : inspired by brane calculi
- Spiking neural systems
- P colonies : set of cells of a bounded capacity, with minimal object processing rules
- Metabolic Systems
- Mobile Membranes

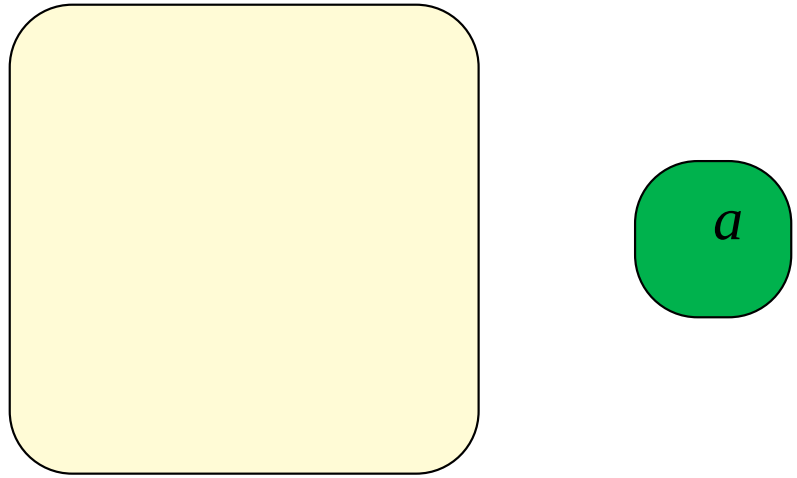
Results

- Characterization of Turing completeness ($RE, \mathbb{N}RE, PsRE$)
- Comparison with Chomsky hierarchy, L-systems
- Polynomial time solutions to NP-complete/PSPACE-somplete problems (using an exponential workspace created in a biological way)
- Other types of mathematical results (normal forms, hierarchies, determinism versus non-determinism, complexity)
- Connections with ambient calculus, petri nets, X-machines, quantum computing, brane calculus
- Simulations and implementations
- Applications

Turing Completeness

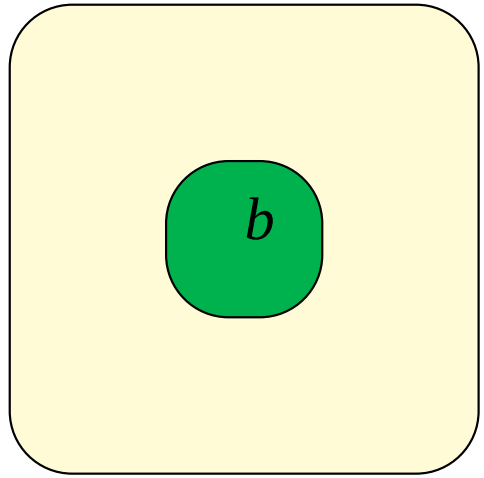
Mobile Membranes

Endocytosis



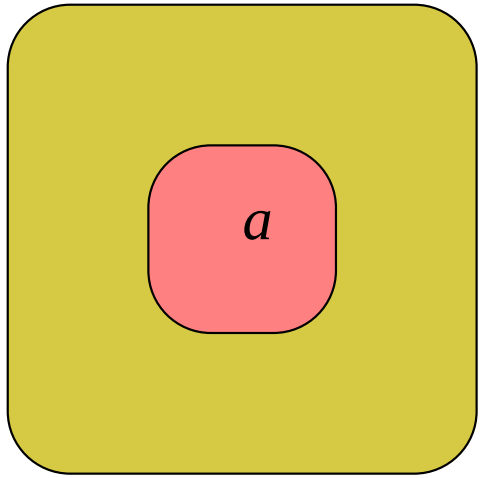
Mobile Membranes

Endocytosis



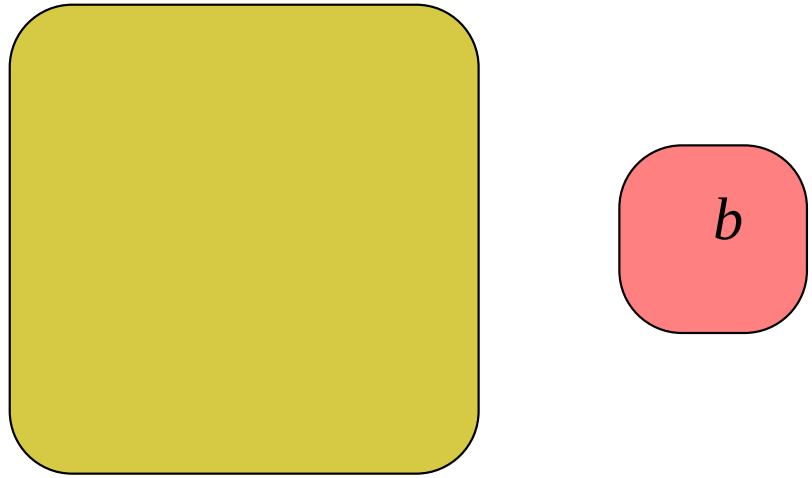
Mobile Membranes

Exocytosis



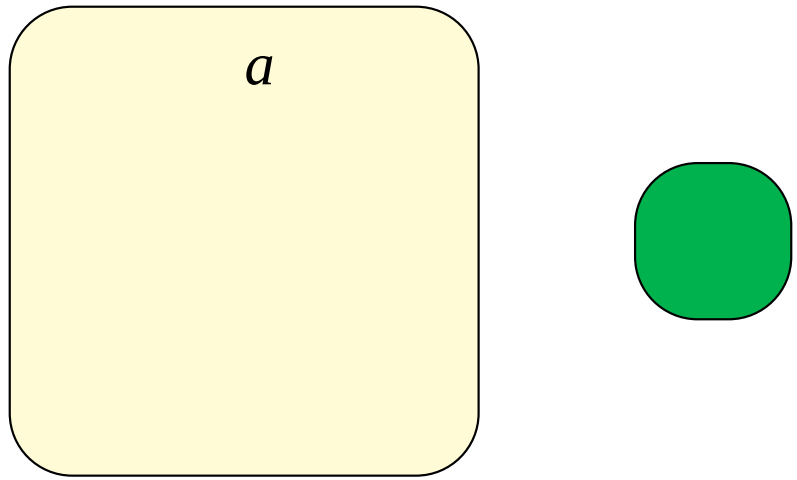
Mobile Membranes

Exocytosis



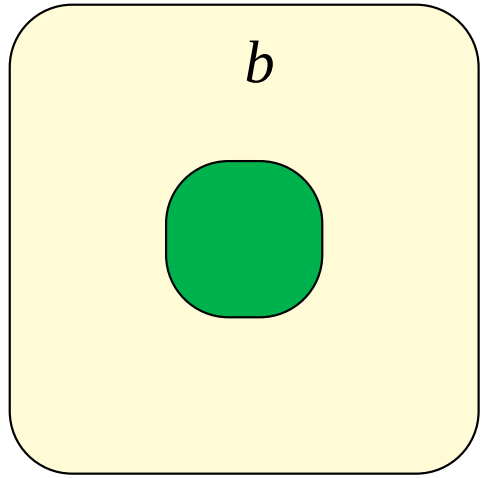
Mobile Membranes

Forced Endocytosis



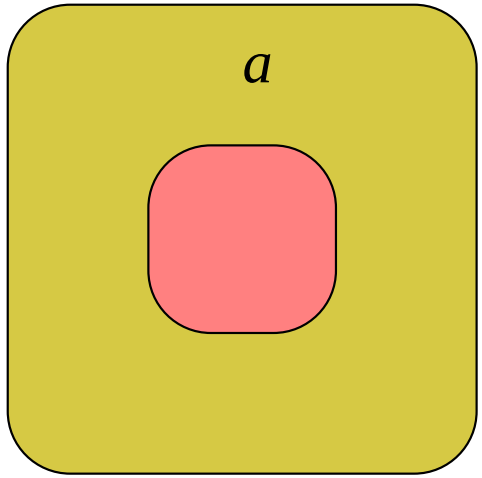
Mobile Membranes

Forced Endocytosis



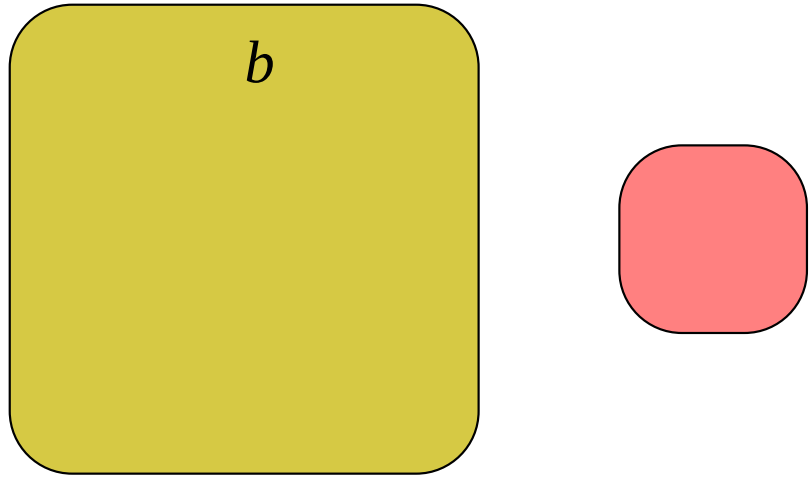
Mobile Membranes

Forced Exocytosis



Mobile Membranes

Forced Exocytosis



Mobile Membranes

$[a]_h[\ ]_m \rightarrow [[w]_h]_m$	endocytosis	(*endo*)
$[[a]_h]_m \rightarrow [w]_h[\ ]_m$	exocytosis	(*exo*)
$[\ ]_h[a]_m \rightarrow [[\ ]_h w]_m$	forced endocytosis	(*fendo*)
$[a[\ ]_h]_m \rightarrow [\ ]_h[w]_m$	forced exocytosis	(*fexo*)

Mobile Membranes are Turing Complete

Families $\mathbb{N}EM_n(\alpha)$, $\alpha \subseteq \{exo, endo, fendo, fexo\}$

Theorem

$\mathbb{N}EM_5(endo, exo, fendo, fexo) = \mathbb{N}RE$

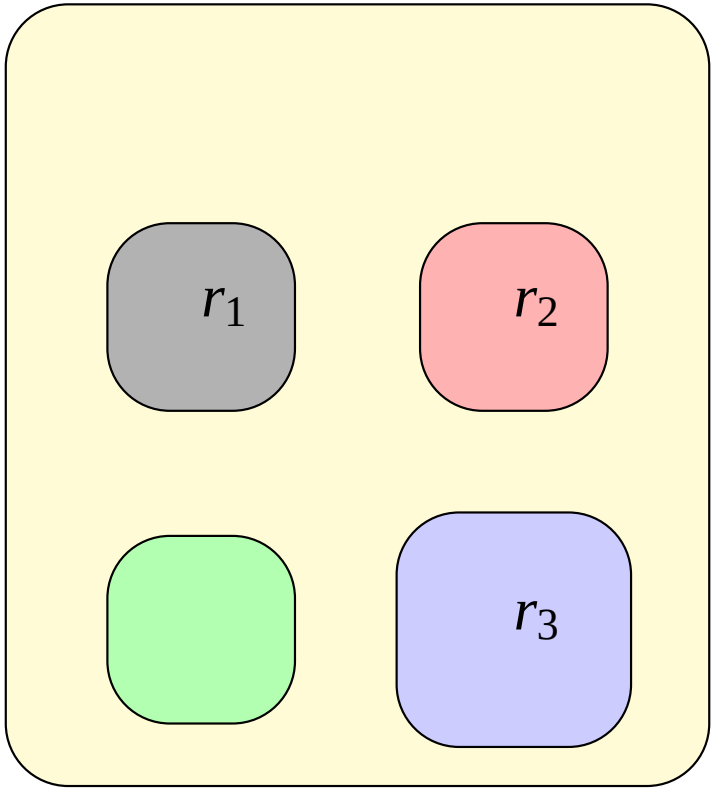
- Proof Idea: Simulate a register machine with 3 registers
- Register machine : (n, P, i, h)
  - n : number of registers
  - P : program with labeled instructions of the form $(add(r), k, l)$ or $(sub(r), k, l)$
  - i : initial instruction; h: final instruction

Theorem

If $L \subseteq V^*$, $card(V) = k, L \in RE$, then a 3-register machine M exists such that for every $w \in V^*$ we have $w \in L$ if and only if M halts when starting with $val_{k+1}(w)$ in its first register; in the halting step, all registers of the machine are empty.

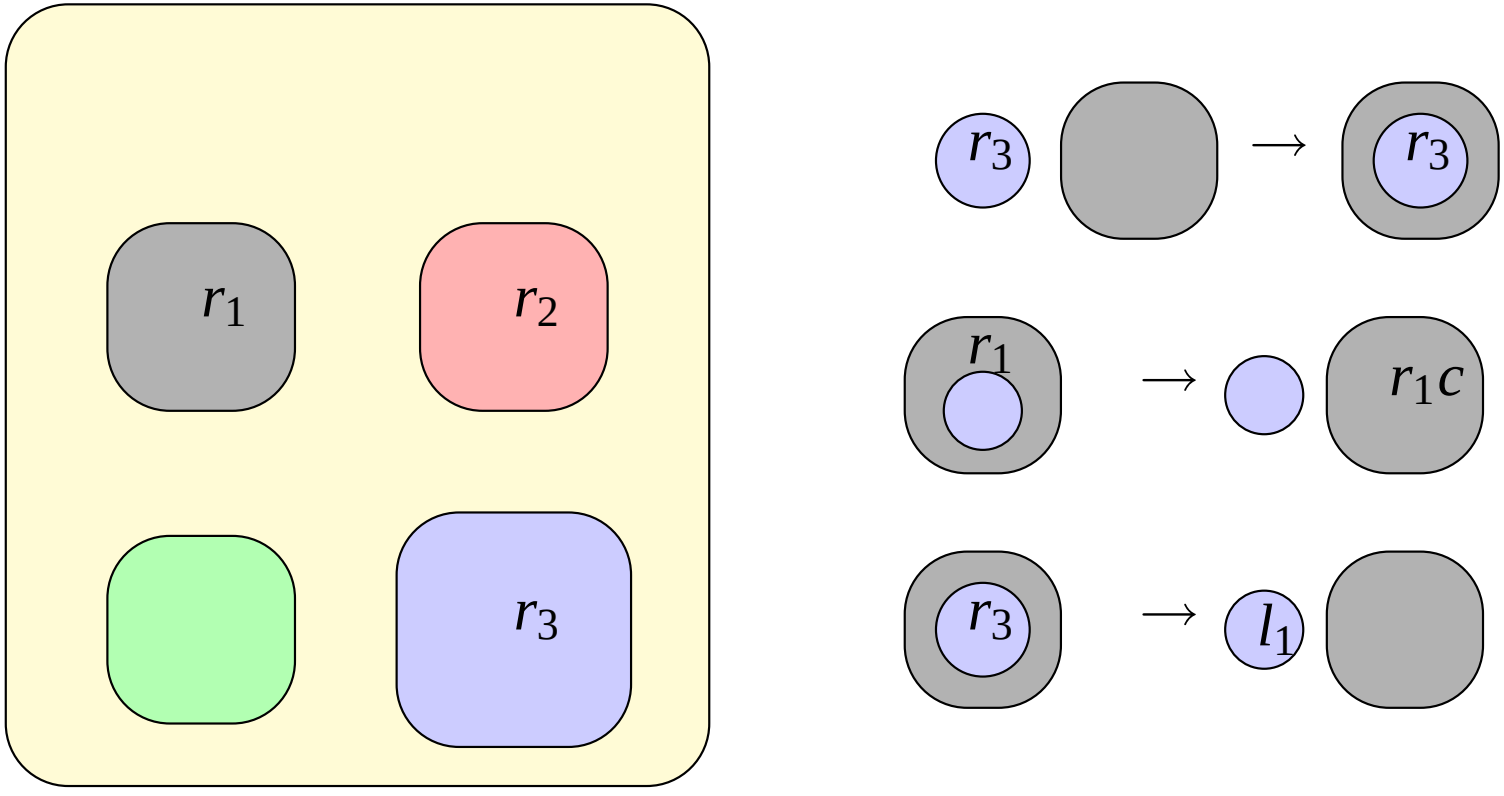
Proof Details

Initial configuration



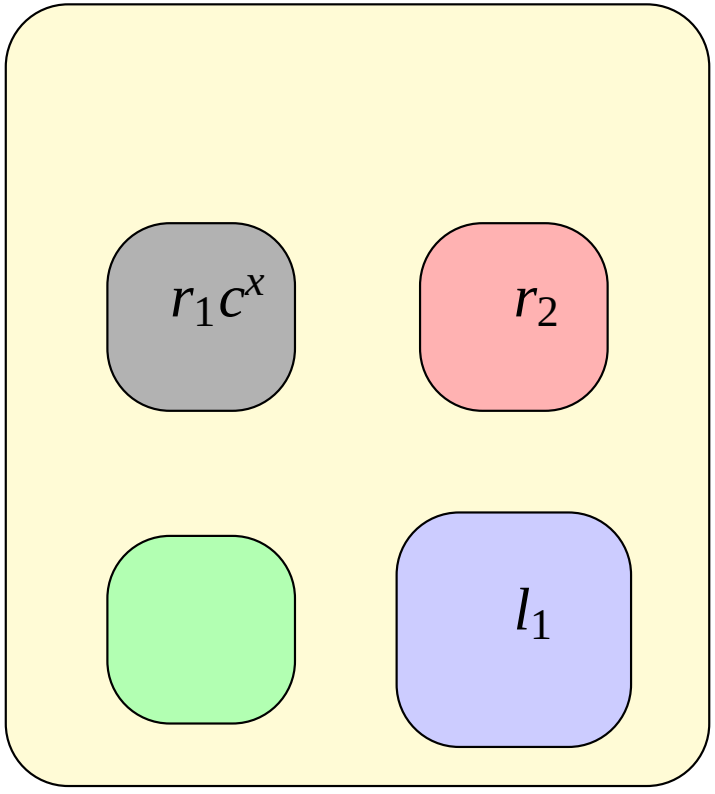
Proof Details

Generation of Initial contents of register 1:



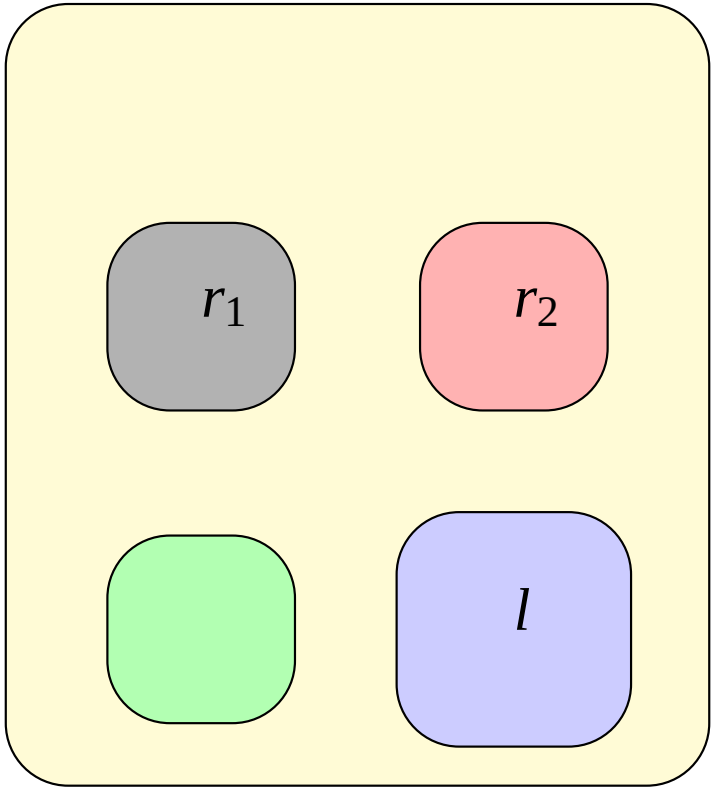
Proof Details

Generation of Initial contents of register 1:



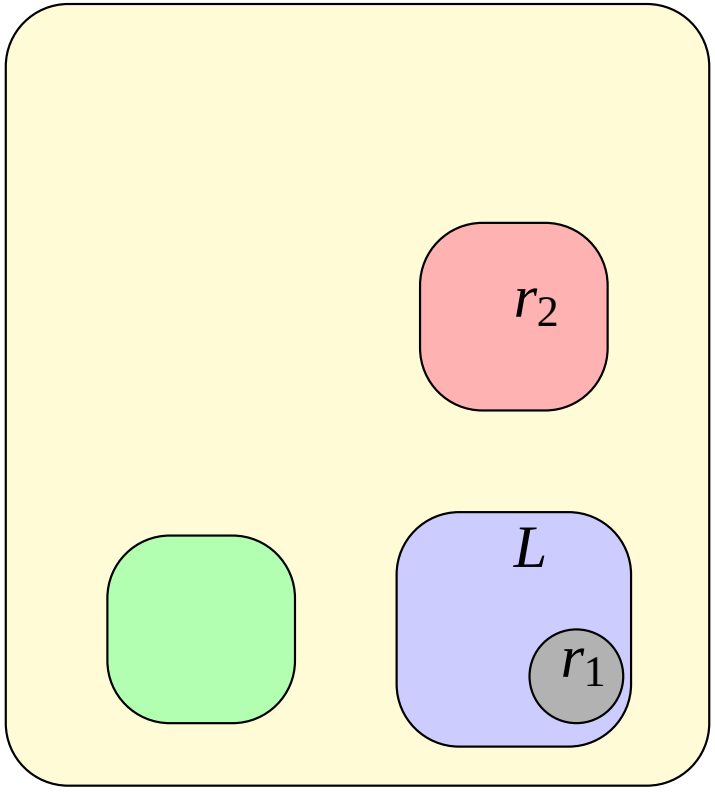
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 empty.



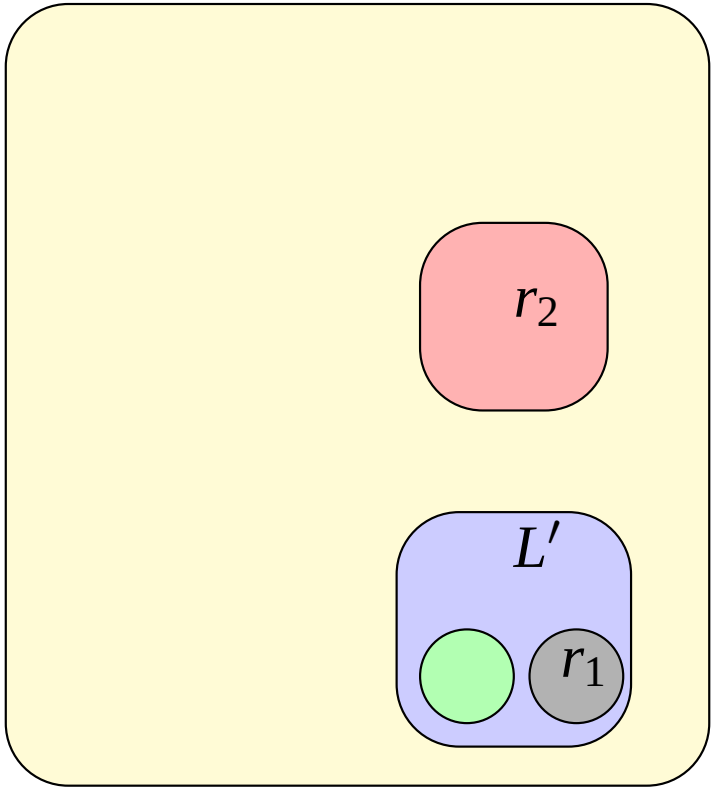
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 empty.



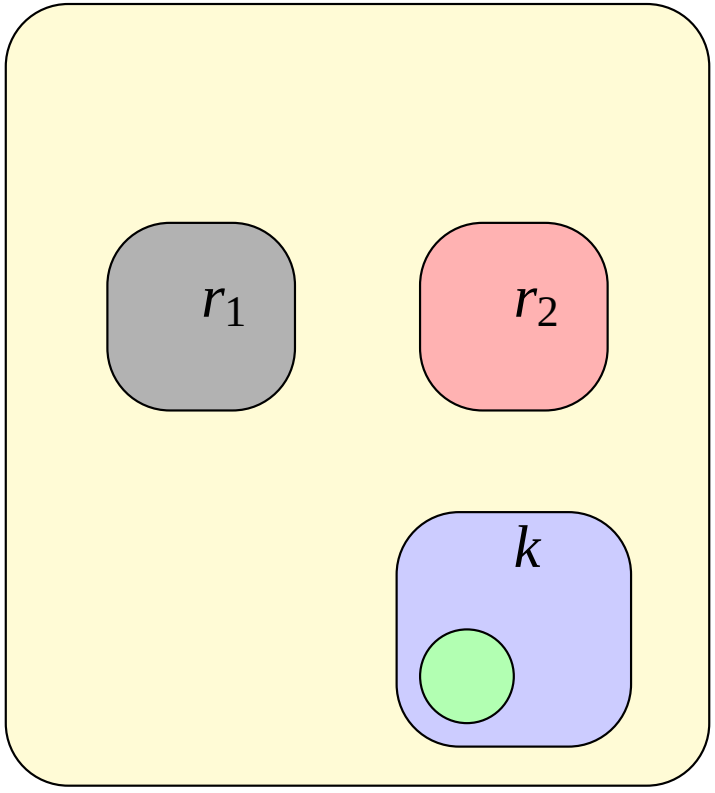
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 empty.



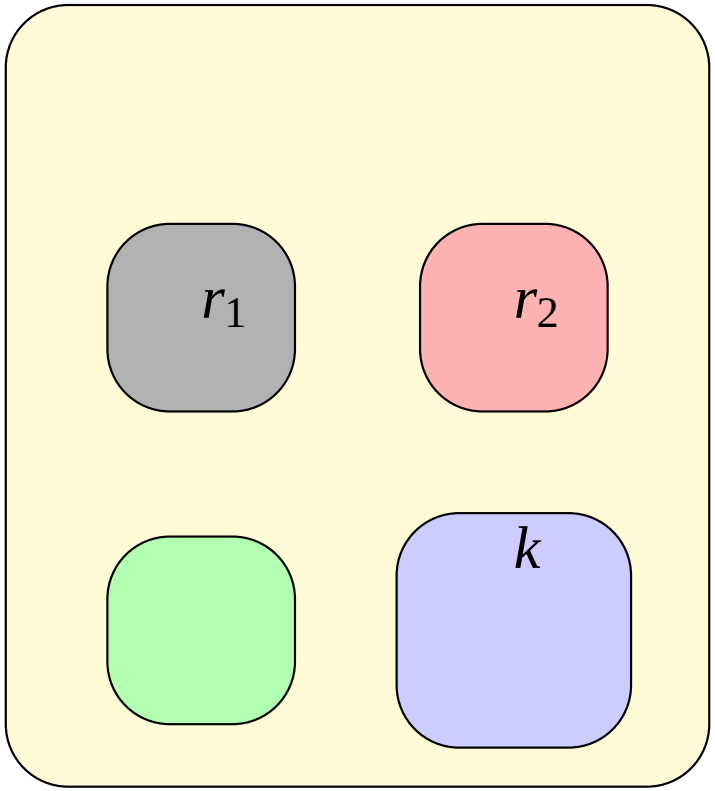
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 empty.



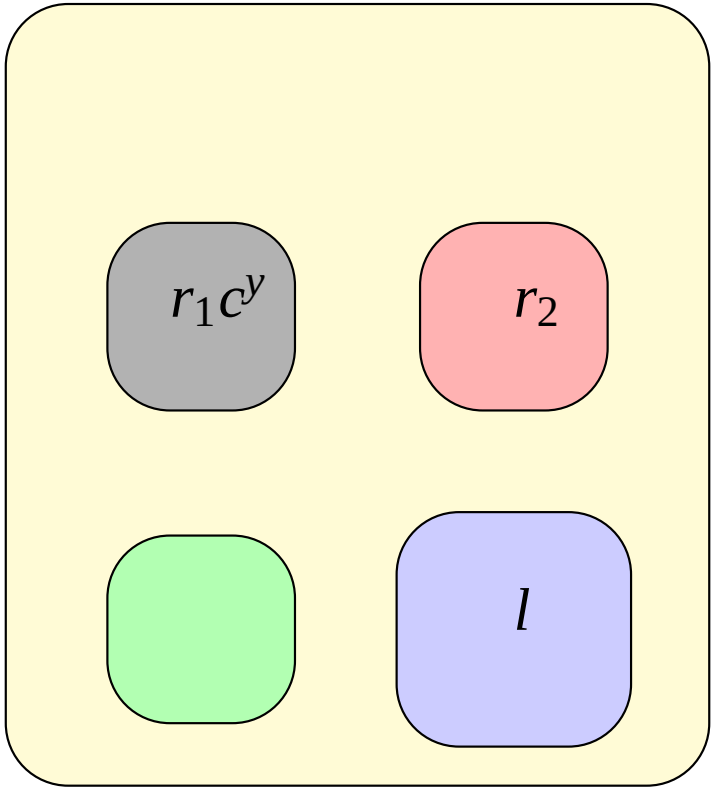
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 empty.



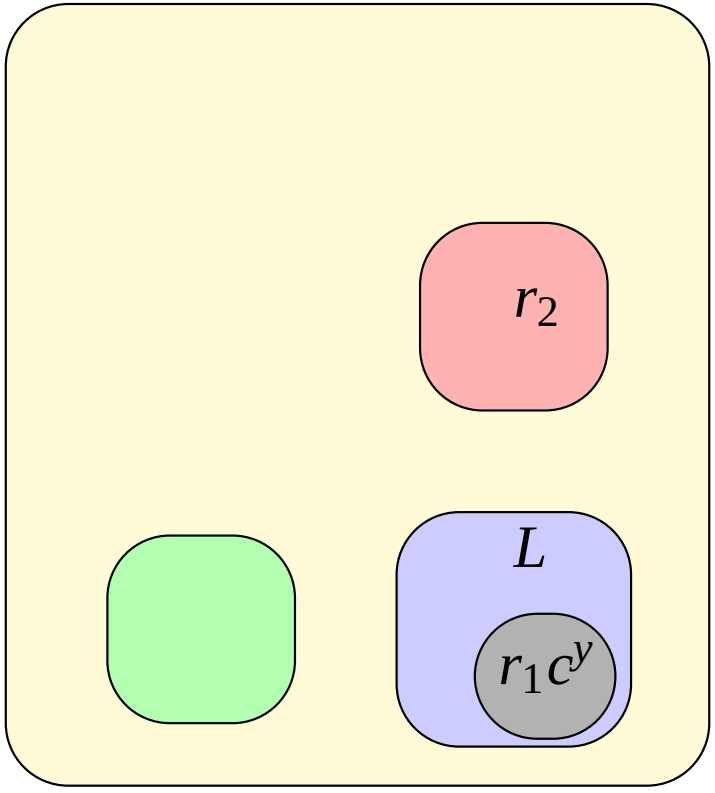
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 non-empty.



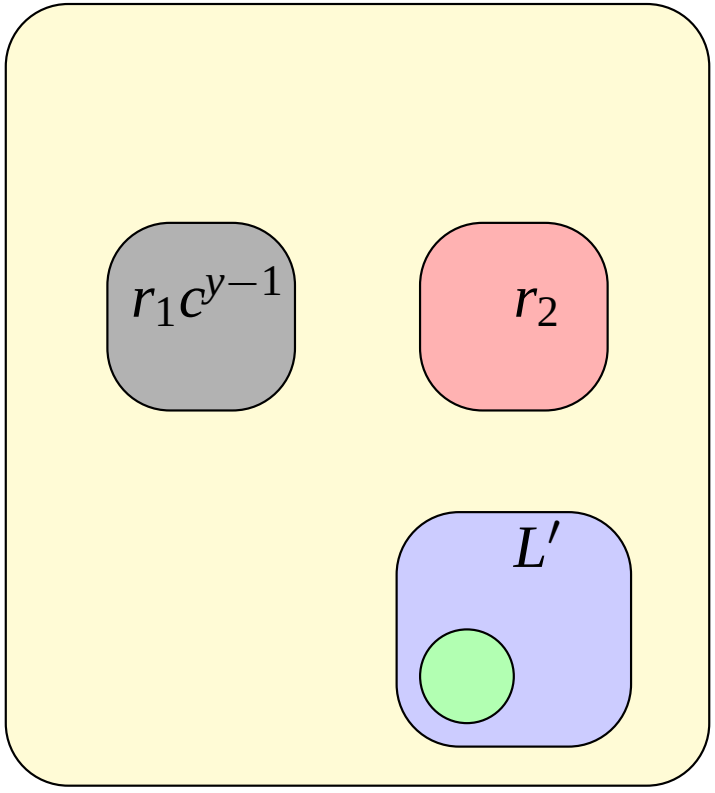
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 non-empty.



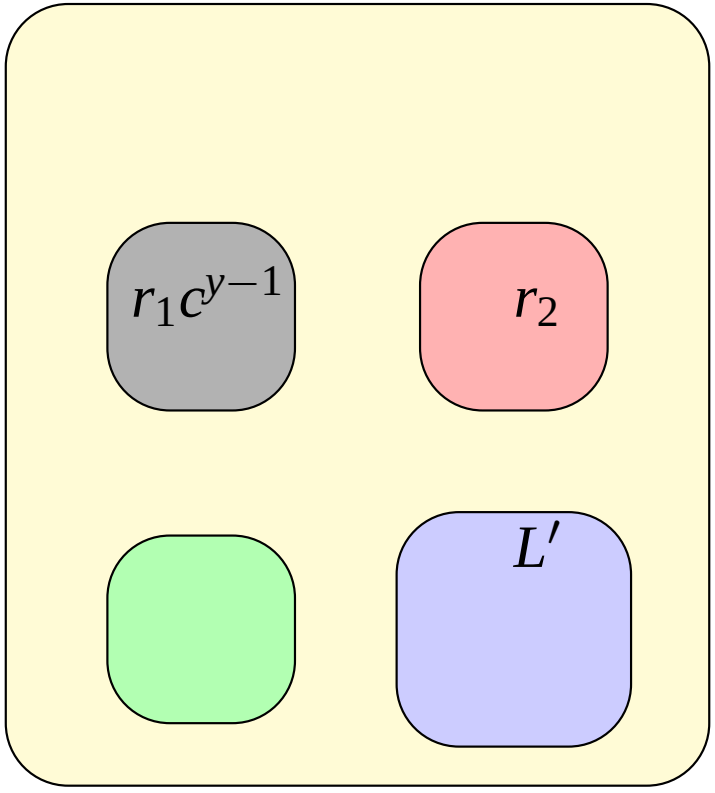
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 non-empty.



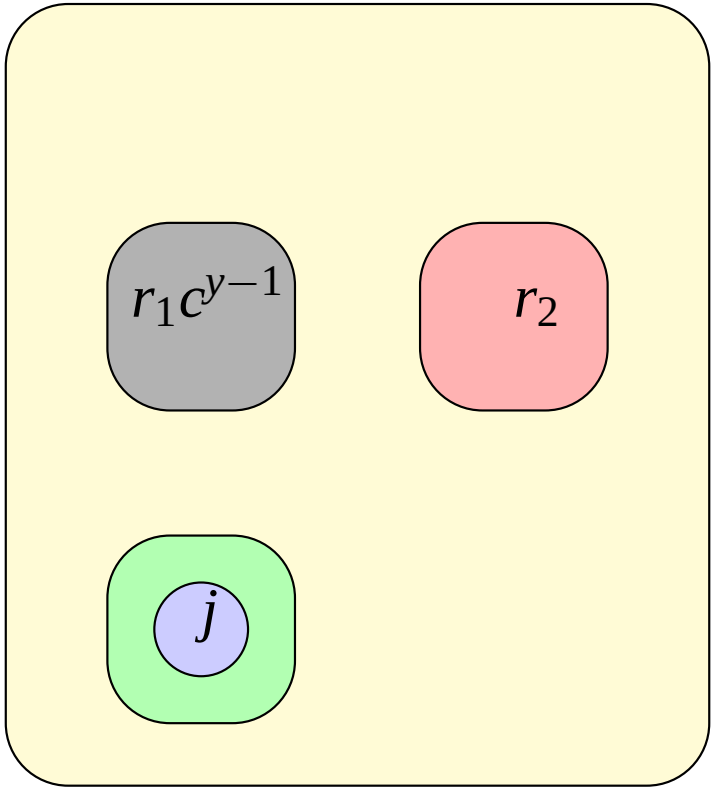
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 non-empty.



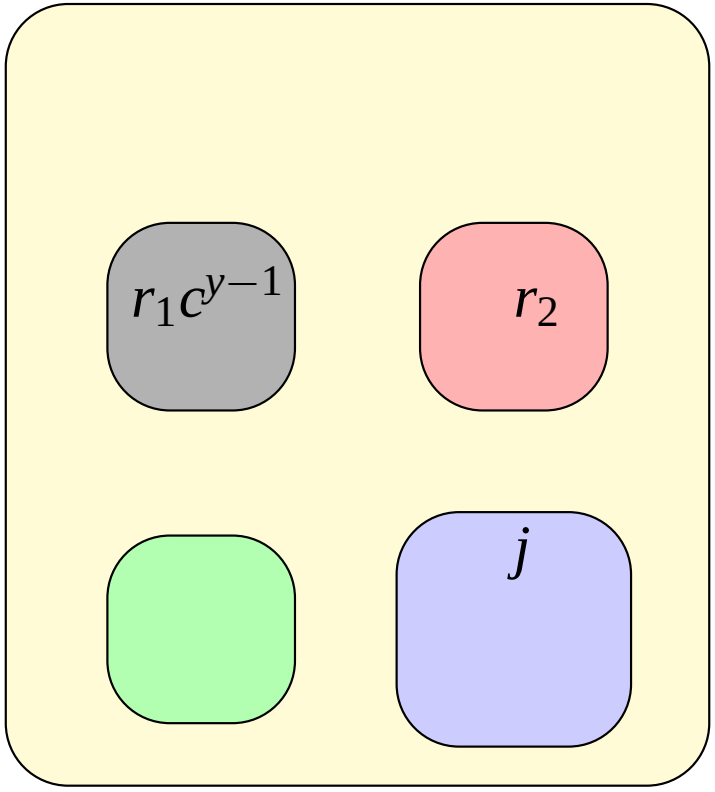
Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 non-empty.



Simulation of a decrement instruction

$l : (sub(1), j, k)$. Register 1 non-empty.



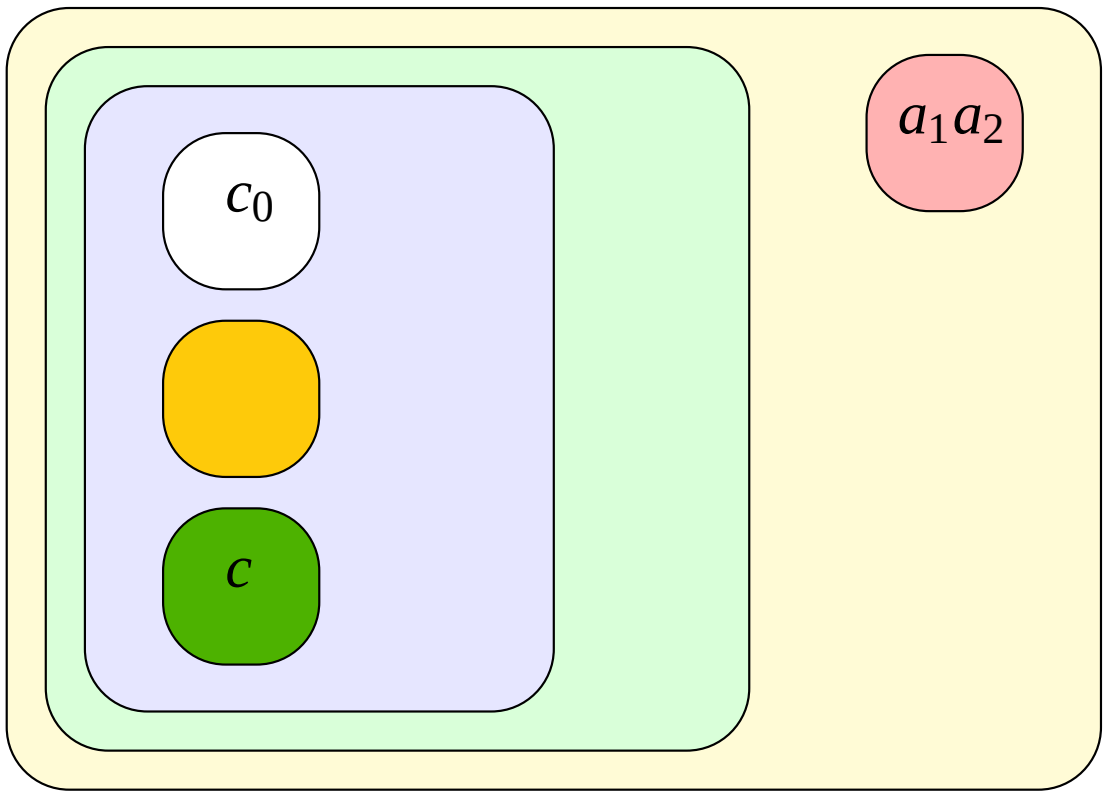
An Application : Uniform Solution to SAT

Solving SAT

- $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ in CNF
- Variables $\{x_1, \ldots, x_n\}$
- C_i of the form $y_1 \vee y_2 \vee \cdots \vee y_r$, $r \leq n$, $y_i = x_i$ or $\neg x_i$
- Propose a polynomial time uniform solution : for all formula instances of size (n, m), takes time $\mathcal{O}(m + n)$
- Uses operations $endo, exo, div$

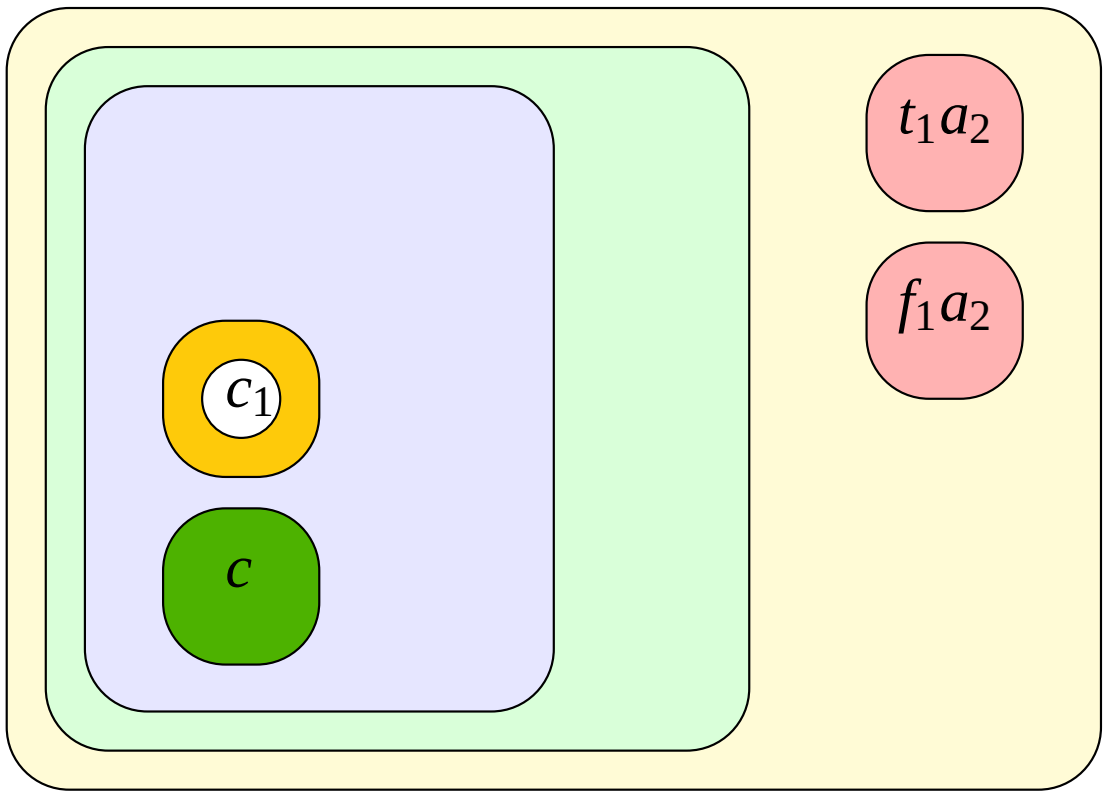
Example

$\varphi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Solution : $\{TT, FF\}$



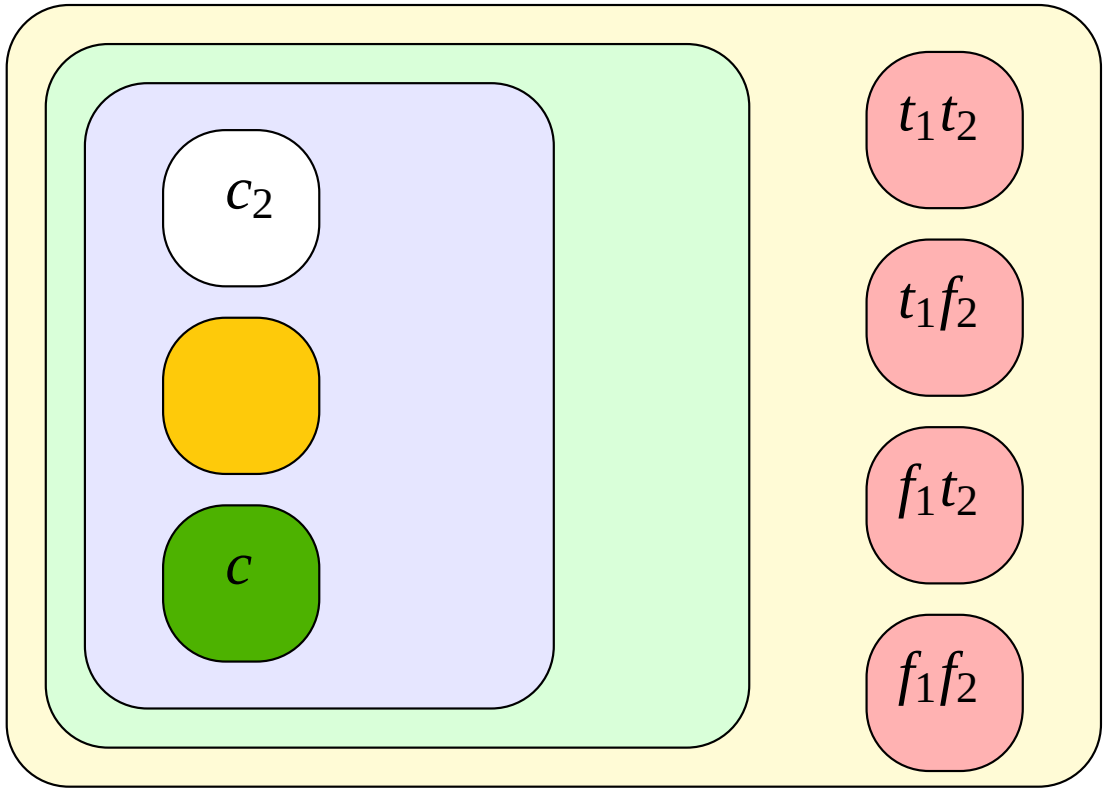
Example

$\varphi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Solution : $\{TT, FF\}$



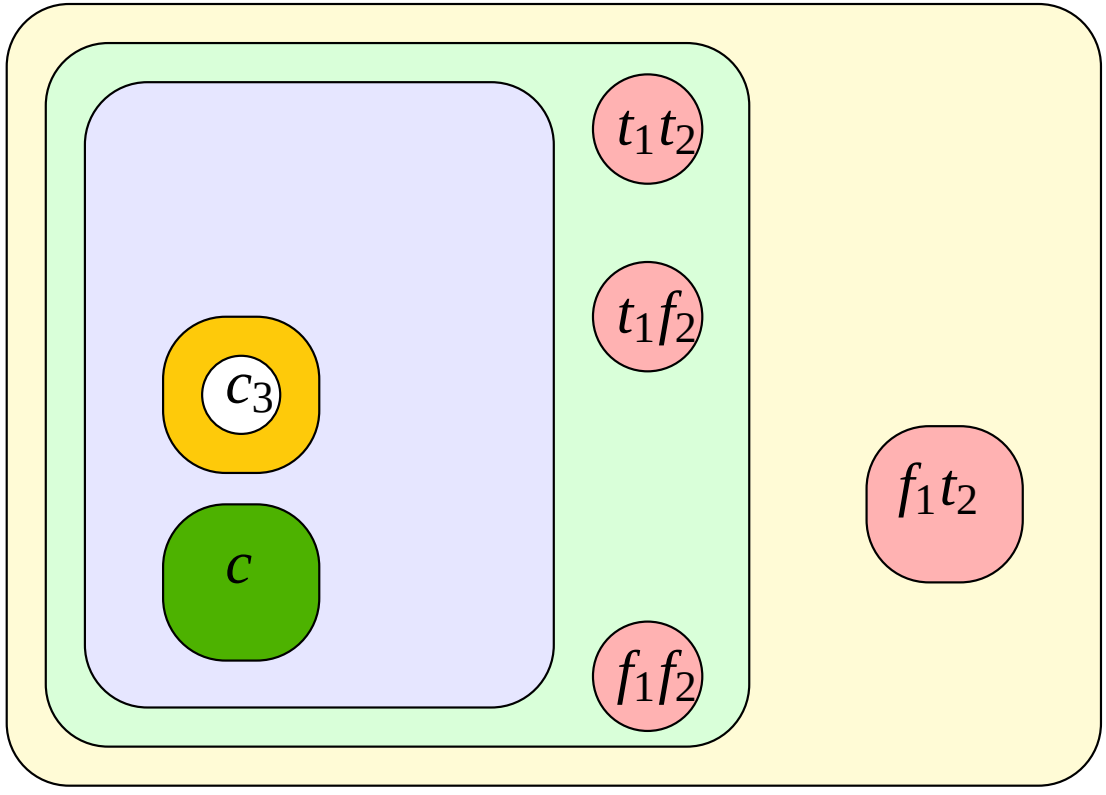
Example

$\varphi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Solution : $\{TT, FF\}$



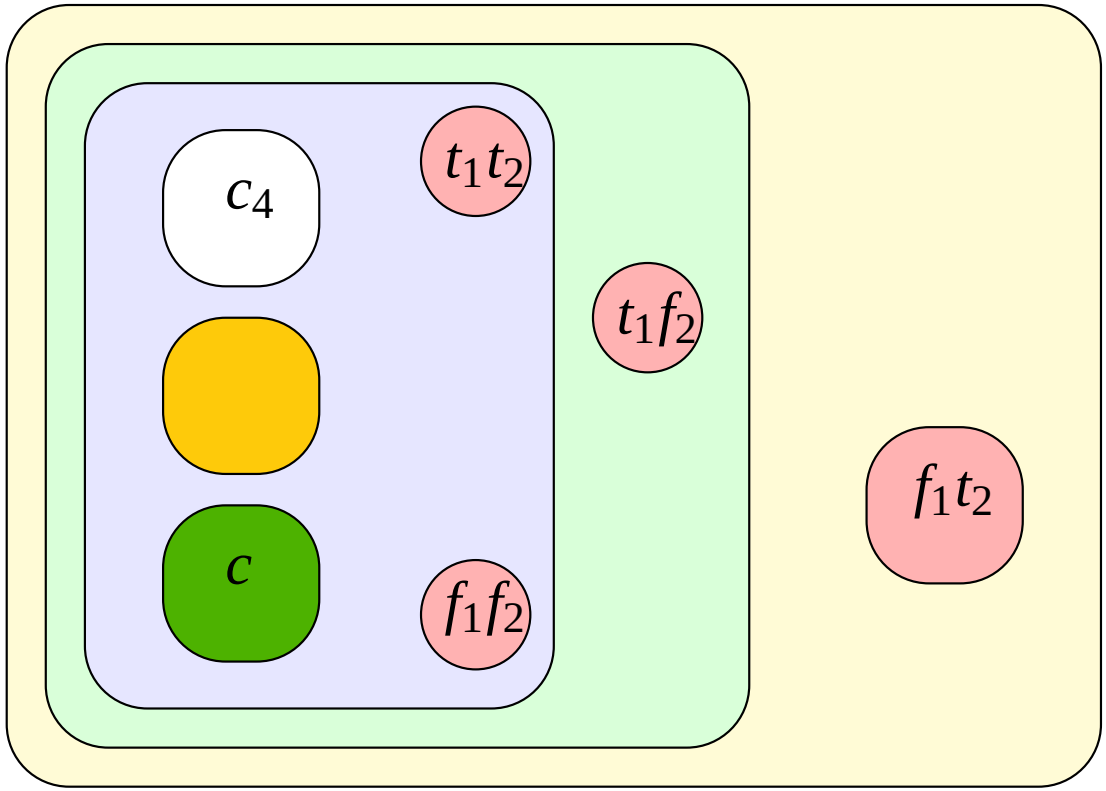
Example

$\varphi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Solution : $\{TT, FF\}$



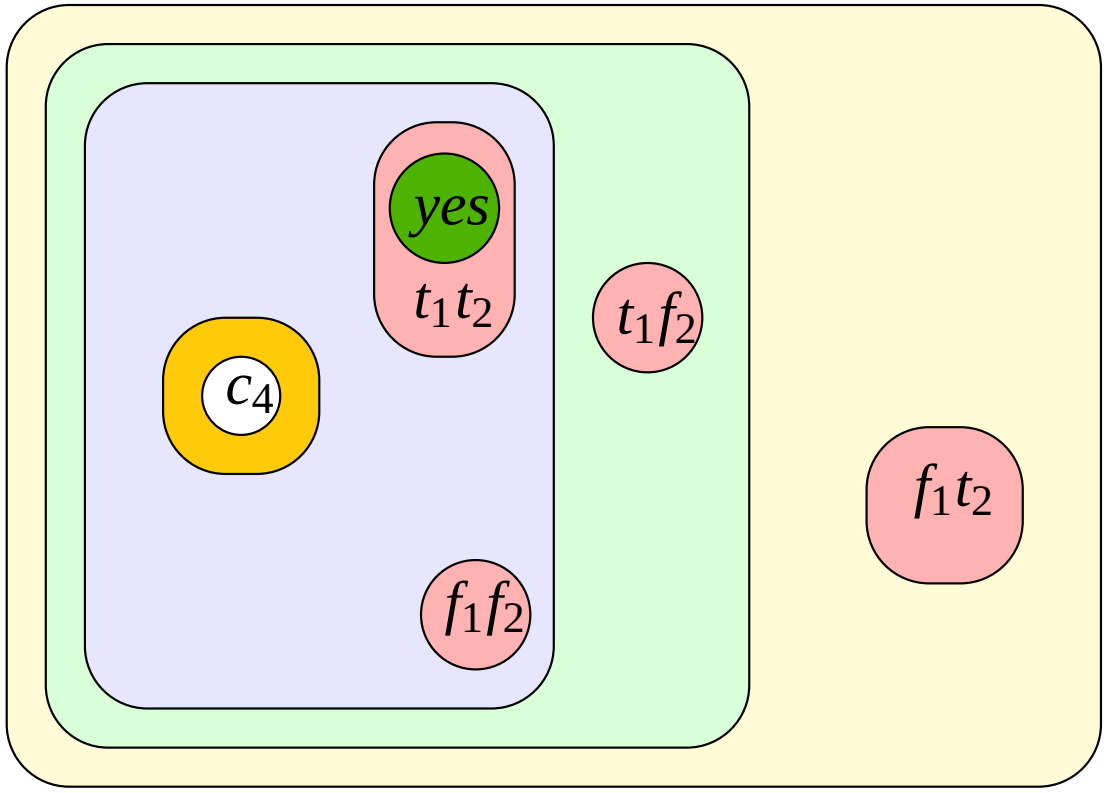
Example

$\varphi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Solution : $\{TT, FF\}$



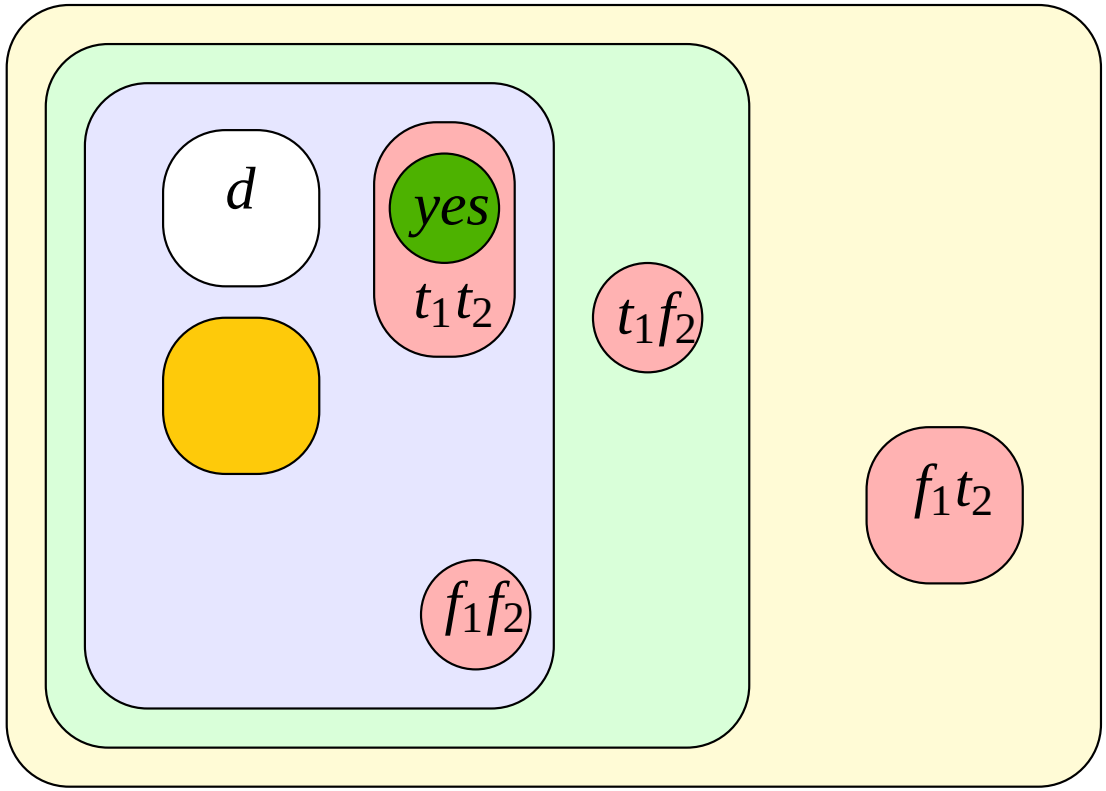
Example

$\varphi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Solution : $\{TT, FF\}$



Example

$\varphi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$. Solution : $\{TT, FF\}$



Applications

- biology, medicine, ecosystems
- computer science
- linguistics (modeling framework/parsing)
- optimization (membrane algorithms)
- economics

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.

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

Implementation Efforts

Currently underway at *E. Keinan lab at Technion, Haifa*. Success would mean a huge boost to the area, and the development of *wet computers*

Open problems, research directions

- 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 versus non-deterministic, complexity classes etc
- semantics (operational, behavioural)
- user friendly, flexible, efficient software for bio-applications
- implementations (electronic, bio-lab)

State of the art

- `http://ppage.psystems.eu`
- *Handbook of Membrane Computing*, Oxford University Press, 2010.

Across the world

Originated in Turku, Finland in Nov 1998. Major active groups

- Bucharest and Iaşi, Romania
- Budapest, Hungary
- Caltech and Santa Barbara, USA
- Edinburgh and Sheffield, UK
- Leiden, The Netherlands
- Madrid and Seville, Spain
- Milan and Verona, Italy
- Chennai and Mumbai, India
- Opava, Czech Republic
- Toyama and Waseda, Japan
- Vienna, Austria

THANKYOU