Brainstorming Week on Membrane Computing (BWMC 2024)
24-26 January 2024, Sevilla Spain

SN P systems with coloured spikes and multiple channels in the rules.

Rodrigo Llanes & José M. Sempere
Valencian Research Institute for Artificial Intelligence (VRAIN)
Department of Information Systems and Computation
Universitat Politècnica de València

jsempere@dsic.upv.es

jsempere.webs.upv.es

VRAIN
Valencian Research Institute for Artificial Intelligence

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Brainstorming Week on Membrane Computing (BWMC 2024)
24-26 January 2024, Sevilla Spain

Use SN P systems with multiple channels in the rules and different kinds of spikes to directly simulate other computacional models inspired by viruses and plasmids

Previous and related works

- Alhazov, A., Freund, R., Oswald, M., Slavkovik, M. (2006). Extended Spiking Neural P Systems. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds) Membrane Computing. WMC 2006. Lecture Notes in Computer Science, vol 4361. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11963516_8

  The authors introduce different target neurons at every rule

- Verlan, S., Freund, R., Alhazov, A. *et al.* A formal framework for spiking neural P systems. *J Membr Comput* **2**, 355–368 (2020). https://doi.org/10.1007/s41965-020-00050-2

  A general framework to introduce different ingredients and variants

- Peng, H., Yang, J., Wang, J., Wang, T., Sun, Z., Song, X., Lou, X. (2017) Spiking neural P systems with multiple channels. Neural Networks. 95 pp. 66–71.

  The authors introduce multiple channels to connect the neurons

- Song, T., Rodríguez-Patón, A., Zheng, P., Zeng, X. (2018) Spiking Neural P Systems With Colored Spikes. IEEE Transactions on Cognitive and Developmental Systems 10(4) pp 1106-1115.

  The authors introduce non-singleton alphabets for spikes.

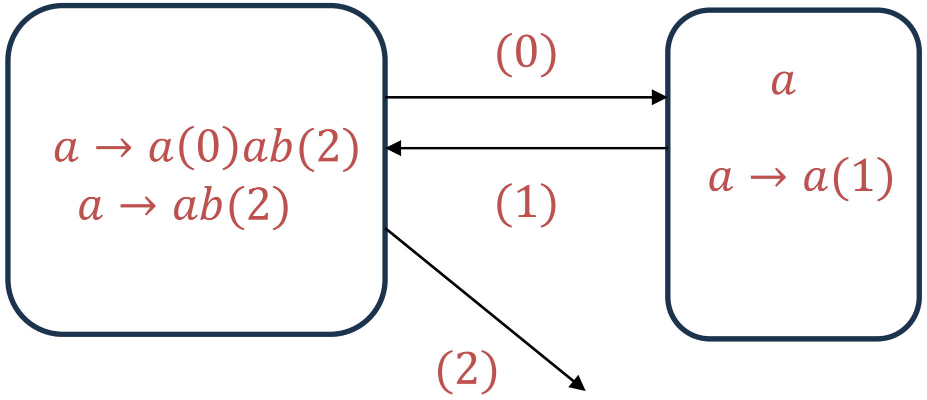
A SN P system with multiple channels and colored spikes (SNP-MC-CS) of degree $m \geq 1$ is defined by $\Pi = (O, L, \sigma_1, \sigma_2, \dots, \sigma_m, syn, in)$, where:

1) $\mathrm{O} = \{\mathrm{a}_1, \mathrm{a}_2, \dots, \mathrm{a}_\mathrm{g}\}$ is an alphabet with g colored spikes
2) $\mathrm{L} = \{1, 2, \dots, \mathrm{N}\}$ is an alphabet of channel labels
3) $\sigma_1, \sigma_2, \dots, \sigma_m$ are neurons of the form $\sigma_i = (w_i, L_i R_i)$, $1 \leq i \leq m$, where:
   a) $w_i \in O^*$ is a string that denotes the initial multiset of spikes in the neuron.
   b) $L_i \subseteq L$ is a finite set of channels labels used in the neuron
   c) R_i is a finite set of rules in the forms:
      1. Firing rules: $E/w_c \rightarrow w_1(l_1)w_2(l_2) \dots w_n(l_n); d$ where E is a regular expression over O, and $w_c, w_i \in O^*$ $1 \leq i \leq n$, $l_i \in L$, $d \geq 0$.
      2. Forgetting rules: $w_c \rightarrow \lambda$, where $w_c \in O^*$.

4) $syn \subseteq \{1, 2, \dots m\} \times \{1, 2, \dots, m, out\} \times L$ are the set of synapse connections with $(i, i, l) \notin syn$ for $1 \leq i \leq m$ and $\forall\, l \in L$. Observe that (i, out, l) denotes that the neuron σ_i sends the spikes out to the environment by the channel l.

5) $in \in \{1, \dots, m\}$ is the input neuron. Observe that the input neuron can be omitted whenever the system is a generator.

An example



The system outputs $\Psi_{\{a,b\}}(\{a^n b^n : n \geq 1\})$

A simulator (I)

```
snp = SNPSystem[int, int]()

snp.set_input(0)

snp.add_symbols(1, *['1']*3)
snp.add_symbols(2, *['1']*2)

snp.add_channel(1, 5, 1)
...
snp.add_channel(5, 2, 5)

snp.add_rule(0, 'a', Multiset(['a']), {2: Multiset(['a'])})
...
snp.add_rule(5, '1*a', Multiset(['a']), {4: Multiset(['a'])})

result = snp.run(['a'], render_steps=True)
```

A simulator (II)

```
input([0])

[1] = {'1'} * 3
[2] = {'1'} * 2

<1> [5] --> [1]
<2> [0] --> [2]
<2> [5] --> [2]
<3> [5] --> [3]
<4> [5] --> [4]
<5> [2] --> [5]

[0] 'a' / {'a'} --> {'a'} <2>

[2] '1'* 'a' / {'1'} --> {'1'} <5>
[2] '1'* 'a' / {'a'} --> {'a'} <5>

[5] '1'* 'a' / {'1'} --> {'1'} <2>, {'1'} <1>
[5] '1'* 'a' / {'a'} --> {'a'} <3>
[5] '1'* 'a' / {'a'} --> {'a'} <4>
```

A simulator (III)

Defining the membrane contents

```
[1] = {'a', 'b'} * 3
```

Defining the channel connections

```
<0> [1] --> [2]
<0> [1] --> out
```

Defining the rules

```
[2] '1'* 'a' / {'a'} --> {'a'} <5> : 1
[2] {'a'} --> {'a'} <5>
{'a'} --> lambda
```

A simulator (IV)

Defining the input membrane

```
input([0])
```

Initializing the membrane contents

```
[1] = {'1'} * 3
```

Tokenizer reply

```
ID(input) OPEN_PARENTHESIS OPEN_MEMBRANE NUMBER(0)
    CLOSE_MEMBRANE CLOSE_PARENTHESIS EOL

OPEN_MEMBRANE NUMBER(1) CLOSE_MEMBRANE EQUAL OPEN_SET
    SYMBOL('1') CLOSE_SET MULT NUMBER(3) EOL

EOF
```

A simulator (V)

Usage

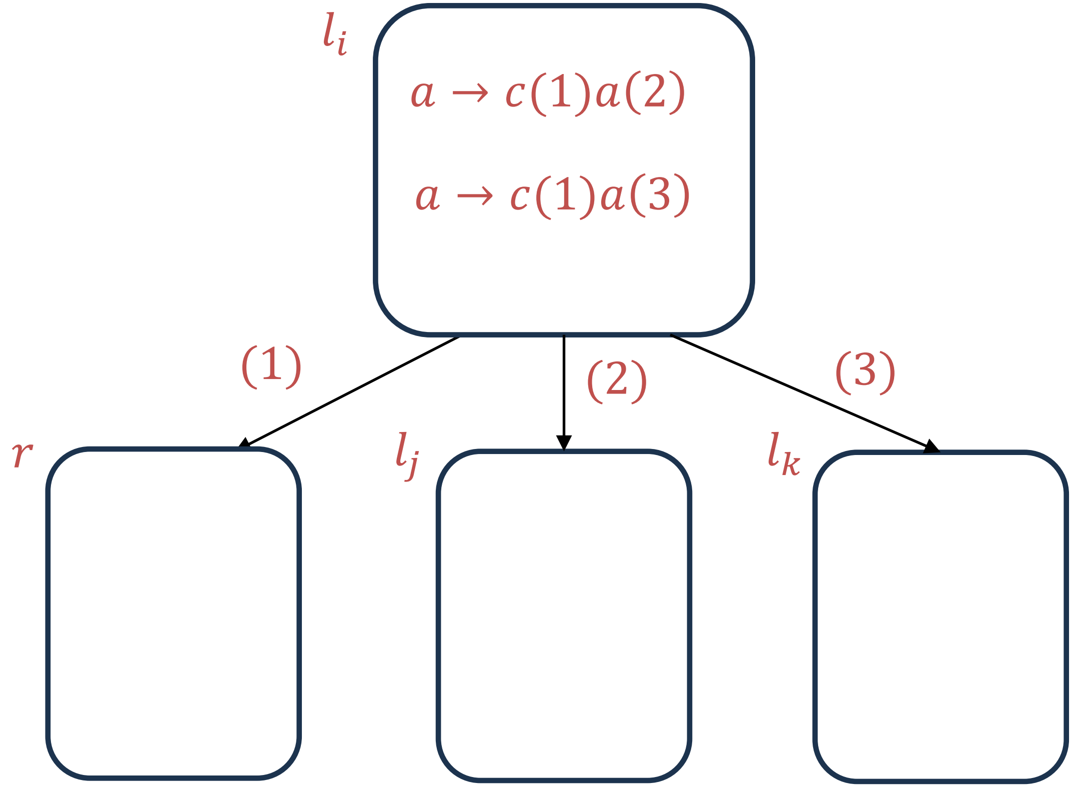
```
Usage: main.py [OPTIONS] SRC

Options:
  -i, --input TEXT
  --separator TEXT
  --no-strip
  --render
  --render-path TEXT
  -r, --repeat INTEGER
  -m, --mode [halt|halt-mc|time|time-mc]
  --max-steps INTEGER
```

Completeness

Simulate register machine instructions: ADD, SUB, HALT

instruction l_i: $(\mathrm{ADD}(r), l_j, l_k)$

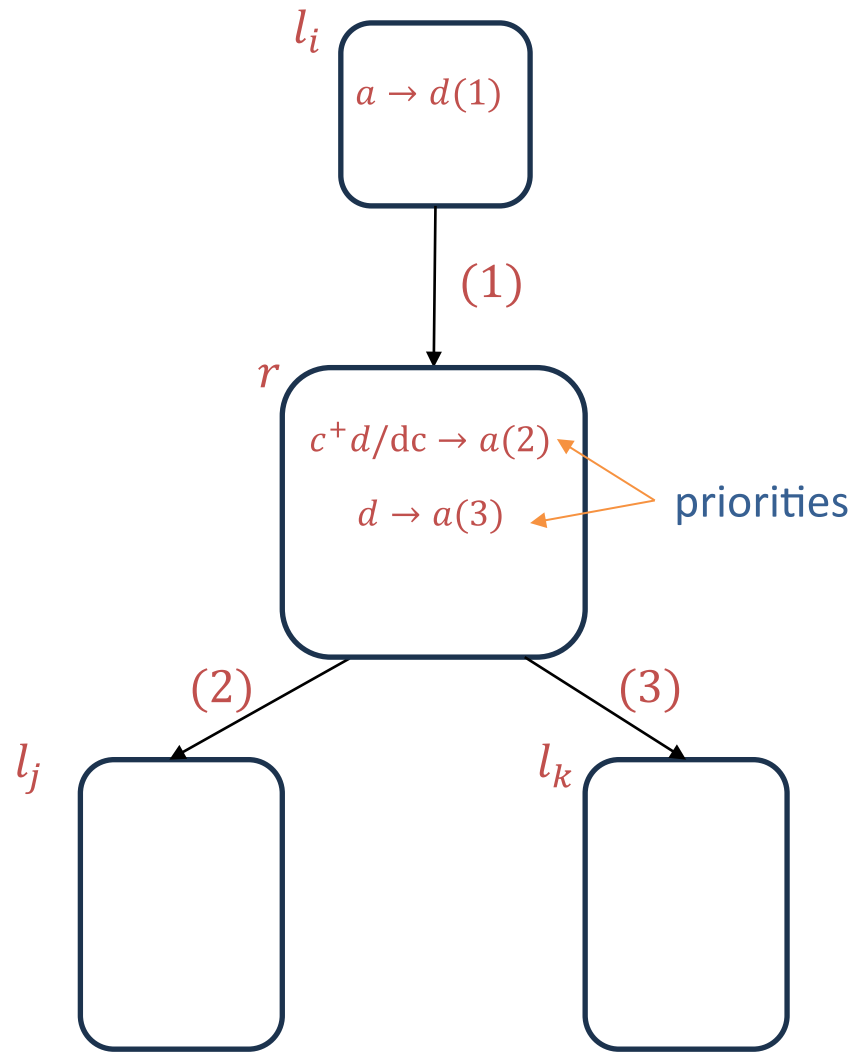


Completeness

Simulate register machine instructions: ADD, SUB, HALT

instruction l_i: $(\mathrm{SUB}(r), l_j, l_k)$

one single SUB instruction



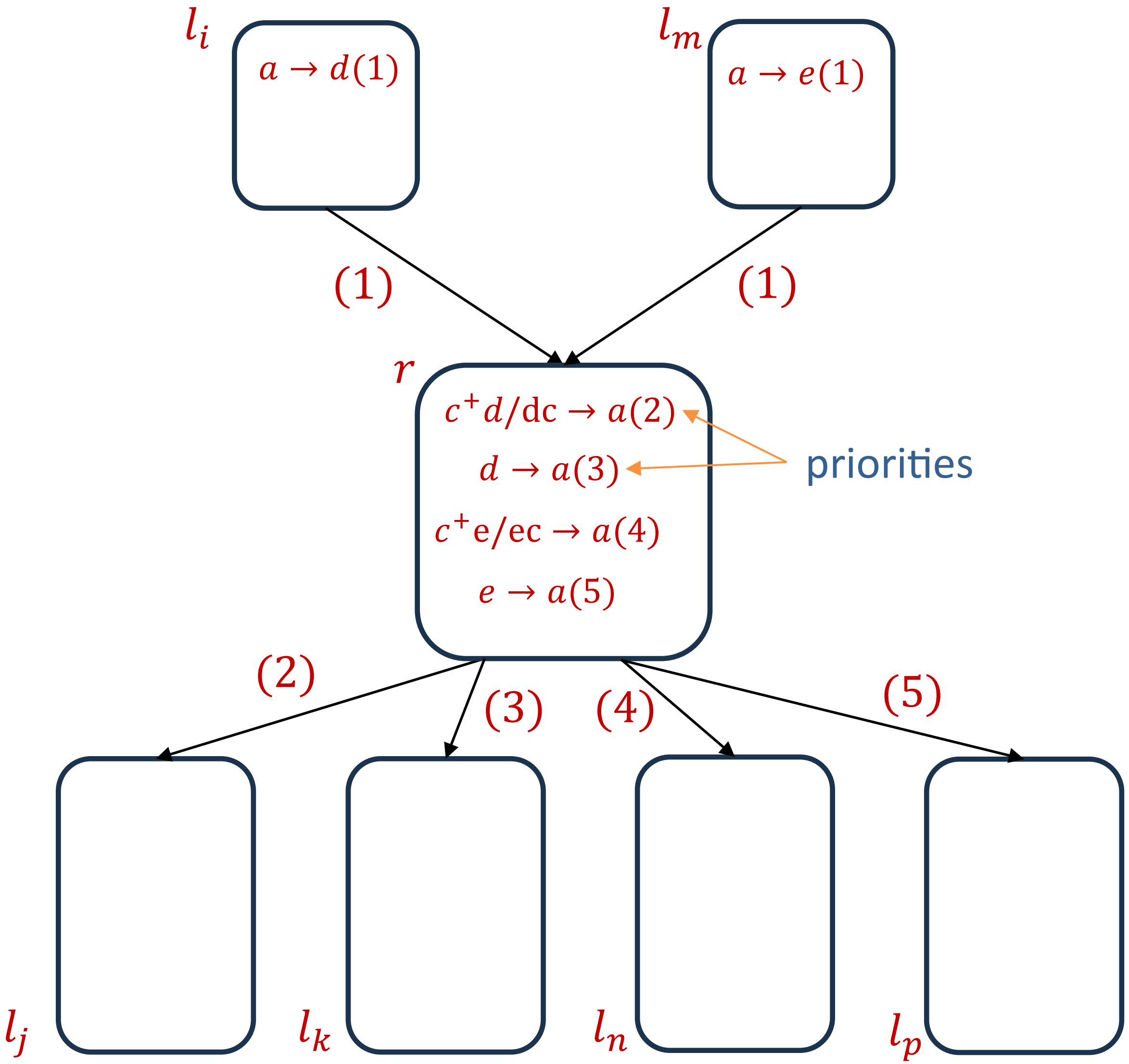
Completeness

Simulate register machine instructions: ADD, SUB, HALT

instruction l_i: $(\mathrm{SUB}(r), l_j, l_k)$

two SUB instructions

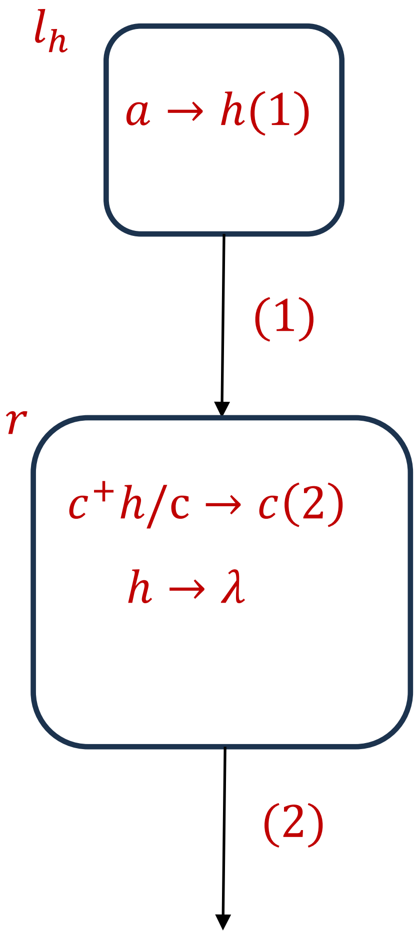
...



Completeness

Simulate register machine instructions: ADD, SUB, HALT

instruction $l_i: HALT$



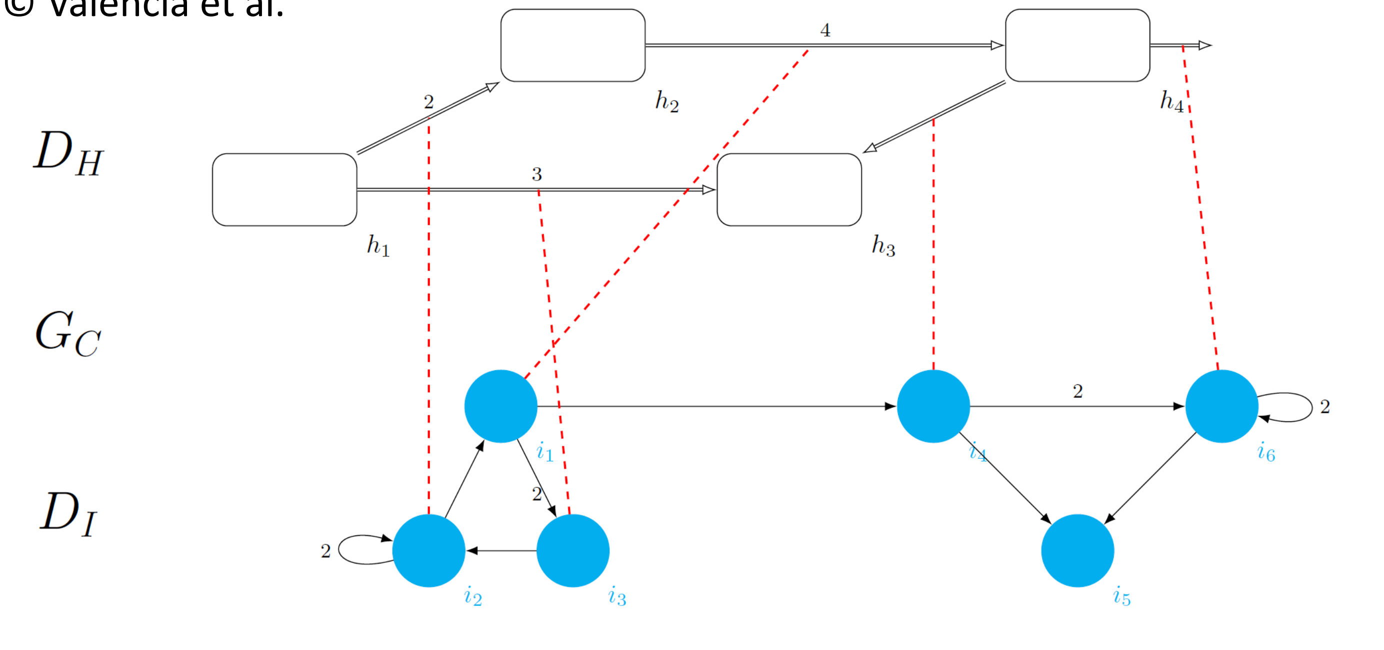
Simulating SN P systems

- Only one single channel c_s
- Only one single spike a_s
- Transform every activation rule

$$E/a^c \rightarrow a; d \quad \Longrightarrow \quad E/a_s^c \rightarrow a_s(c_s); d$$

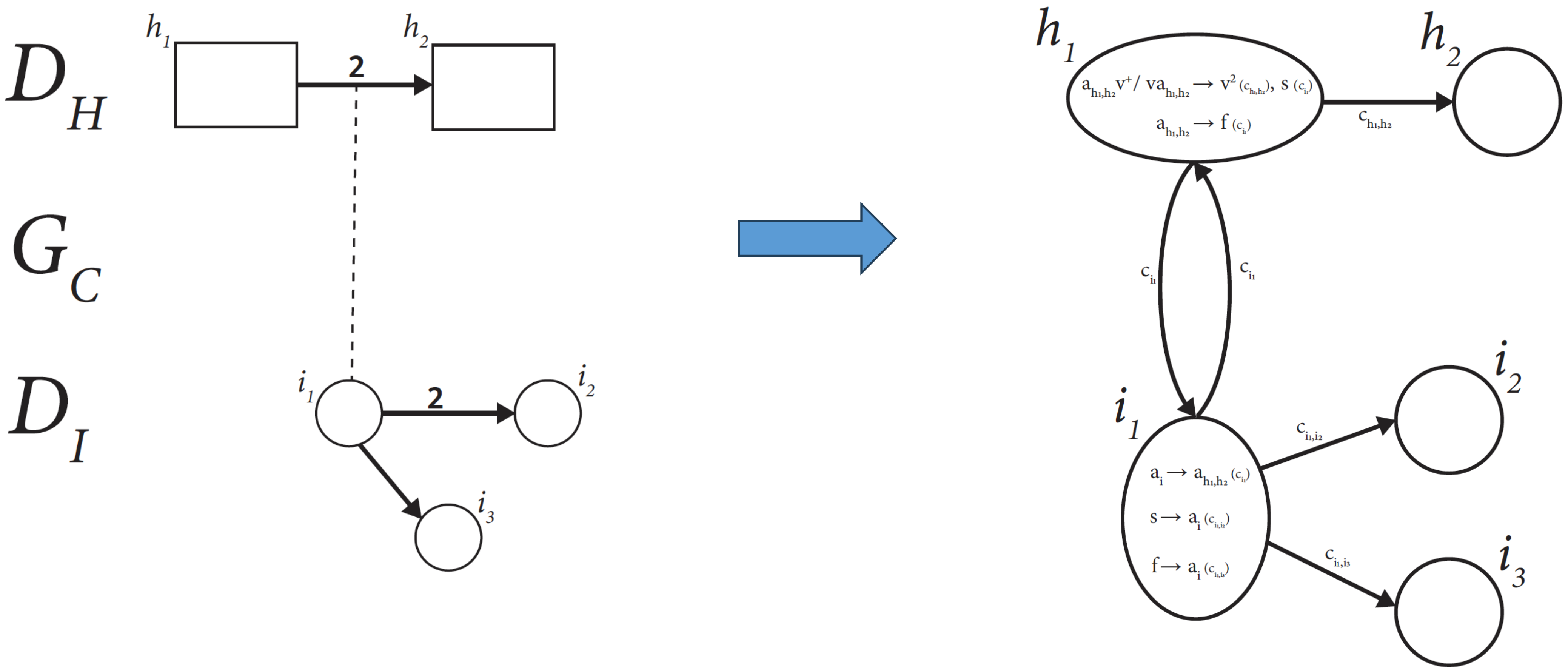
Simulating virus machines

L. Valencia, M.J. Pérez-Jiménez, X. Chen, B. Wang, X. Zheng. Basic virus machines. In J.M. Sempere and C. Zandron (eds) Proceedings of the 16th International Conference on Membrane Computing (CMC16), 17-21 August, 2015, Valencia, Spain, pp. 323-342.



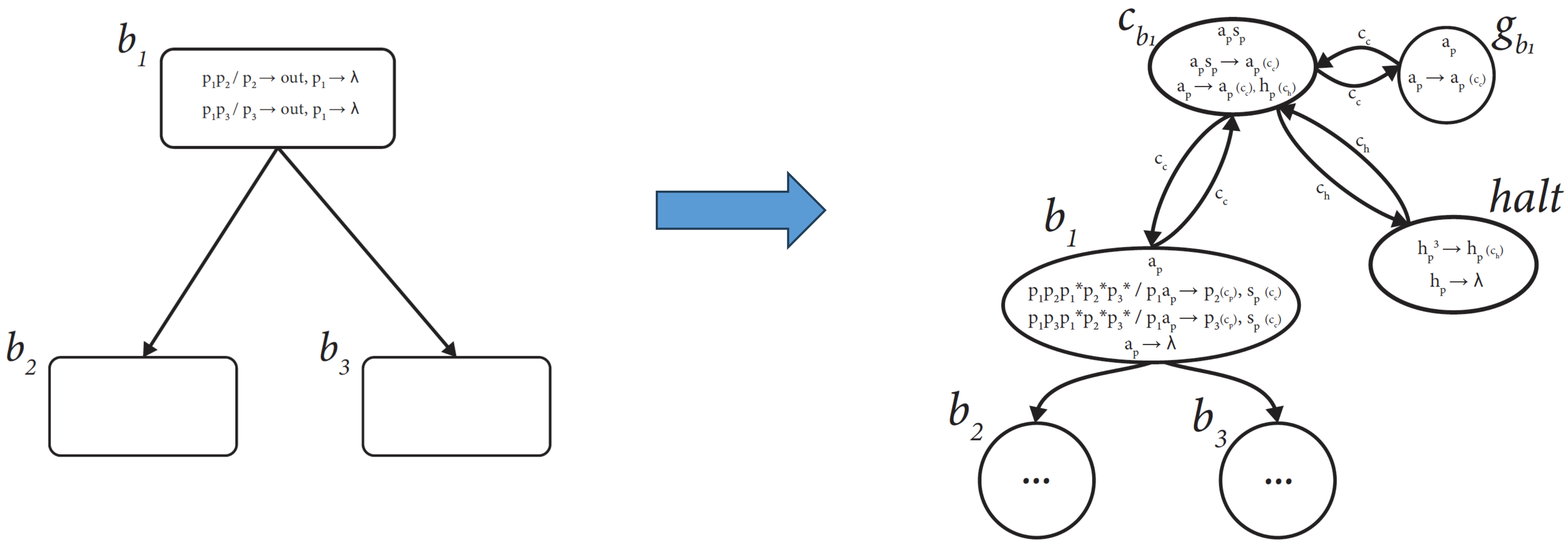
Two kind of channels: channels for behavior (viruses transmition and instructions) and channels for regulation

Simulating virus machines



Simulating computing with plasmids

Y. Li, B. Song and X. Zeng. Neural-Like P Systems With Plasmids and Multiple Channels. IEEE Transactions on NanoBioscience, vol. 22, no. 2, (2023) 420-429



questions ?