On Generalized Communicating P Systems with One Symbol

Erzsébet Csuhaj-Varjú, **György Vaszil**

Computer and Automation Research Institute, Budapest

Sergey Verlan

Université Paris Est, Paris

Generalized Communicating P Systems

Inspired by the **problem** how to define a **common generalization of various purely communicating models** in the framework of **P systems**

[S. Verlan, F. Bernardini, M. Gheorghe, M. Margenstern, 2008]

Interaction rules

$\boxed{(a,i)(b,j) \to (a,k)(b,l)}$ with a,b objects and i,j,k,l cell labels

Such an interaction rule may be applied if there is an object a in cell i and an object b in cell j

As the result of the application of r, the object a moves from cell i to cell k and b moves from cell j to cell l

The rules are classified based on the labels

$\boxed{(a,i)(b,j) \to (a,k)(b,l)}$ with a, b objects and i, j, k, l cell labels

1. $(a,i)(b,i) \to (a,i)(b,j)$ conditional-uniport-out
2. $(a,i)(b,j) \to (a,i)(b,i)$ conditional-uniport-in
3. $(a,i)(b,i) \to (a,j)(b,j)$ symport2
4. $(a,i)(b,j) \to (a,j)(b,i)$ antiport1
5. $(a,i)(b,j) \to (a,i)(b,k)$ presence-move
6. $(a,i)(b,i) \to (a,j)(b,k)$ split
7. $(a,i)(b,j) \to (a,k)(b,k)$ join
8. $(a,i)(b,j) \to (a,k)(b,i)$ chain
9. $(a,i)(b,j) \to (a,k)(b,l)$ parallel-shift

Outline

- Generalized communicating P systems, **definitions, results**
  - **related results**
  - known results on their generative **power** with different **rule-type combinations**

- Generalized communicating P systems with minimal interaction, known **results**

  \- **one type** of rules

- **Our topic**: Generalized communicating P systems with

  \- **one type** of rules + one **single object**

Generalized communicating P systems, GCPS-s

$$\Pi = (O, E, w_1, \ldots, w_n, R, h)$$

1. O is a finite **set of objects**
2. $E \subseteq O$ is the set of **environmental objects**
3. $w_i \in O^*$, $1 \leq i \leq n$, are the **initial cell contents**
4. R is a finite set of **interaction rules** $(a,i)(b,j) \to (a,k)(b,l)$, $a, b \in O$, $0 \leq i, j, k, l \leq n$
5. $h \in \{1, \ldots, n\}$, is the **output cell**.

Configurations, transitions, result

Configuration: $(z_0, z_1, \ldots, z_n)$ with $z_0 \in (O \setminus E)^*$, $z_i \in O^*$, $1 \leq i \leq n$, **halting** if no rule can be applied

Initial configuration: $(\lambda, w_1, \ldots, w_n)$

Transition: $(z_0, z_1, \ldots, z_n) \Longrightarrow (z'_0, z'_1, \ldots, z'_n)$, the rules are applied in a **maximal parallel** manner, a computation starts in the initial configuration

Result: the (size of the) **multiset** of objects present inside the **output cell** at the end of the computation

The **set of non-negative integers generated** by a GCPS Π in this way is denoted by $N(\Pi)$

Related models, results, history

1. P systems with **minimal symport/antiport** are **computationally complete** systems, **Petri nets**, the relationship of Petri nets and P systems, topology of information flow

2. Conditional-uniport-in + conditional-uniport-out is **computationally complete**

   [Verlan, Bernardini, Gheorghe, Margenstern, 2006]

3. GCP systems, **several variants** and rule type combinations are **computationally complete**

   [Verlan, Bernardini, Gheorghe, Margenstern, 2008]

4. GCPMI systems, using **one rule type** are **computationally complete**, except for antiport1 (these languages are finite)

   [Csuhaj-Varjú, Verlan, to appear]

Restricted Variants

$\boxed{(a,i)(b,j) \to (a,k)(b,l)}$ with a, b objects and i, j, k, l cell labels

1. $(a,i)(b,i) \to (a,i)(b,j)$ conditional-uniport-out
2. $(a,i)(b,j) \to (a,i)(b,i)$ conditional-uniport-in
3. $(a,i)(b,i) \to (a,j)(b,j)$ symport2
4. $(a,i)(b,j) \to (a,j)(b,i)$ antiport1
5. $(a,i)(b,j) \to (a,i)(b,k)$ presence-move
6. $(a,i)(b,i) \to (a,j)(b,k)$ split
7. $(a,i)(b,j) \to (a,k)(b,k)$ join
8. $(a,i)(b,j) \to (a,k)(b,i)$ chain
9. $(a,i)(b,j) \to (a,k)(b,l)$ parallel-shift

GCP systems with one type of rules $\rightarrow$ GCPSMI

Generalized communicating P system **with minimal interaction** are computationally complete:

Type	**Number of cells needed**
uin	30
$uout$	30
$sym2$	10
$split$	9
$join$	7
$presence$	36
$chain$	not bounded
$shift$	19

[Verlan, Bernardini, Gheorghe, Margenstern, 2008]
[Csuhaj-Varjú, Verlan, to appear]

Proof techniques of the GCPS and GCPSMI results

- Direct simulation of a **register machine**

- Simulating **functional blocks** which implement the instructions of a register machine

  - uniport block, main block, zero block

[Verlan, Bernardini, Gheorghe, Margenstern, 2006]

Register machines

$$M = (Q, R, q_0, q_f, P)$$

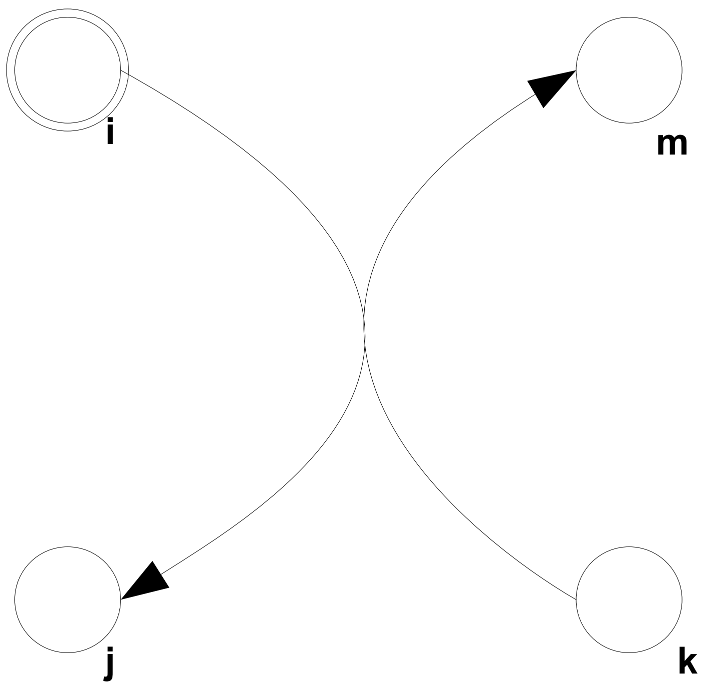
1. Q is the set of **states**
2. $R = \{A_1, \ldots, A_k\}$, $k \geq 1$, is a set of **registers**
3. $q_0 \in Q$ is the **initial state**, $q_f \in Q$ is the **final state**
4. P is a set of **instructions** of the following forms:
   (a) $(p, A+, q, s)$, where $p, q, s \in Q, p \neq q_f, A \in R$, called an *increment instruction*,
   (b) $(p, A-, q, s)$, where $p, q, s \in Q, p \neq q_f, A \in R$, called a *decrement instruction*.

   Furthermore, for every $p \in Q$, $(p \neq q_f)$, there is exactly one instruction of the form either $(p, A+, q, s)$ or $(p, A-, q, s)$.

Uniport Block

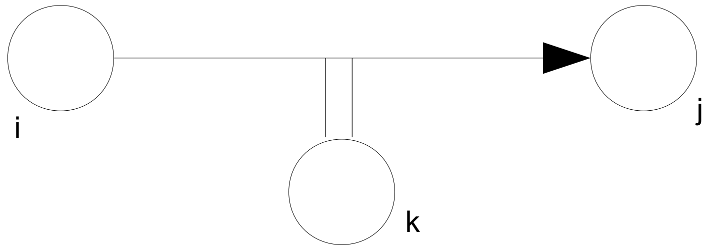
It implements **a uniport rule** $(X, i) \rightarrow (X, j)$.

The main block



A symbol in i (while moving to j) triggers the movement of a symbol from k to m. If k is empty, an infinite loop occurs.

The zero block



A symbol appearing in i moves to j if k is empty. If k is not empty, an infinite loop occurs.

Our results: GCPSMI with one symbol

Does it make sense with one symbol?

1. $(\bullet, i)(\bullet, i) \rightarrow (\bullet, i)(\bullet, j)$ conditional-uniport-out - NO
2. $(\bullet, i)(\bullet, j) \rightarrow (\bullet, i)(\bullet, i)$ conditional-uniport-in - YES
3. $(\bullet, i)(\bullet, i) \rightarrow (\bullet, j)(\bullet, j)$ symport2 - NO
4. $(\bullet, i)(\bullet, j) \rightarrow (\bullet, j)(\bullet, i)$ antiport1 - NO
5. $(\bullet, i)(\bullet, j) \rightarrow (\bullet, i)(\bullet, k)$ presence-move - YES
6. $(\bullet, i)(\bullet, i) \rightarrow (\bullet, j)(\bullet, k)$ split - NO
7. $(\bullet, i)(\bullet, j) \rightarrow (\bullet, k)(\bullet, k)$ join - YES
8. $(\bullet, i)(\bullet, j) \rightarrow (\bullet, k)(\bullet, i)$ chain - YES
9. $(\bullet, i)(\bullet, j) \rightarrow (\bullet, k)(\bullet, l)$ parallel-shift - YES

Our results: GCPSMI with one symbol

$$\boxed{\text{GCPSMI with one symbol} + \textbf{parallel-shift} = \mathbb{N}RE}$$

Increment instruction: $(p, R+, q, s)$:

$(1^{(p)}) : (\bullet, p)(\bullet, 0) \to (\bullet, q)(\bullet, A_i)$ $\quad$ $(2^{(p)}) : (\bullet, p)(\bullet, 0) \to (\bullet, s)(\bullet, A_i)$

Decrement instruction $(p, R-, q, s)$:

$(1^{(p)}) : (\bullet, p)(\bullet, 0) \to (\bullet, 1_p)(\bullet, 2_p)$ $\quad$ $(2^{(p)}) : (\bullet, 1_p)(\bullet, A_i) \to (\bullet, 3_p)(\bullet, 0)$

$(3^{(p)}) : (\bullet, 2_p)(\bullet, 0) \to (\bullet, 4_p)(\bullet, 5_p)$ $\quad$ $(4^{(p)}) : (\bullet, 1_p)(\bullet, 4_p) \to (\bullet, s)(\bullet, 0)$

$(5^{(p)}) : (\bullet, 3_p)(\bullet, 4_p) \to (\bullet, q)(\bullet, 0)$

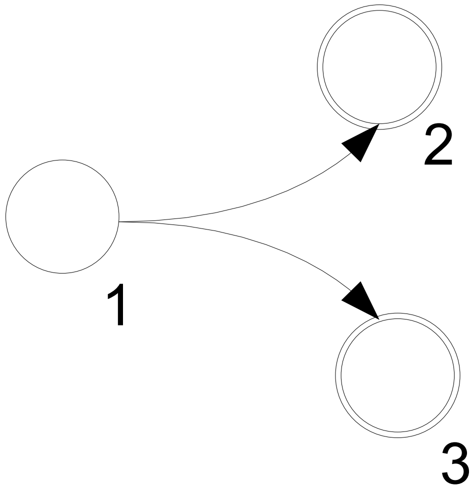
Results continued...

$\boxed{\text{GCPSMI with one symbol} + \textbf{join} = \mathbb{N}RE}$

We implement **parallel-shift rules** in **three phases**.

1. Simulate a **pseudo-split** element
2. **Combine several pseudo-split elements** in order to simulate **split operations**
3. **Combine join and split** operations to simulate **parallel-shift operations**

Pseudo-split



Splits two tokens from 1 into 2 and 3, if the following **condition** holds: In the next step, only one symbol can leave from 2 and 3, or no symbol can leave from at least one of 2, 3.

The **pseudo-split** is implemented as follows (nodes $4, 4'$ have 2 tokens):

$(1) : (\bullet, 1)(\bullet, 4) \to (\bullet, 5)(\bullet, 5)$ $(2) : (\bullet, 1)(\bullet, 4') \to (\bullet, 5')(\bullet, 5')$

$(3) : (\bullet, 4)(\bullet, 5) \to (\bullet, 3)(\bullet, 3)$ $(4) : (\bullet, 4')(\bullet, 5') \to (\bullet, 2)(\bullet, 2)$

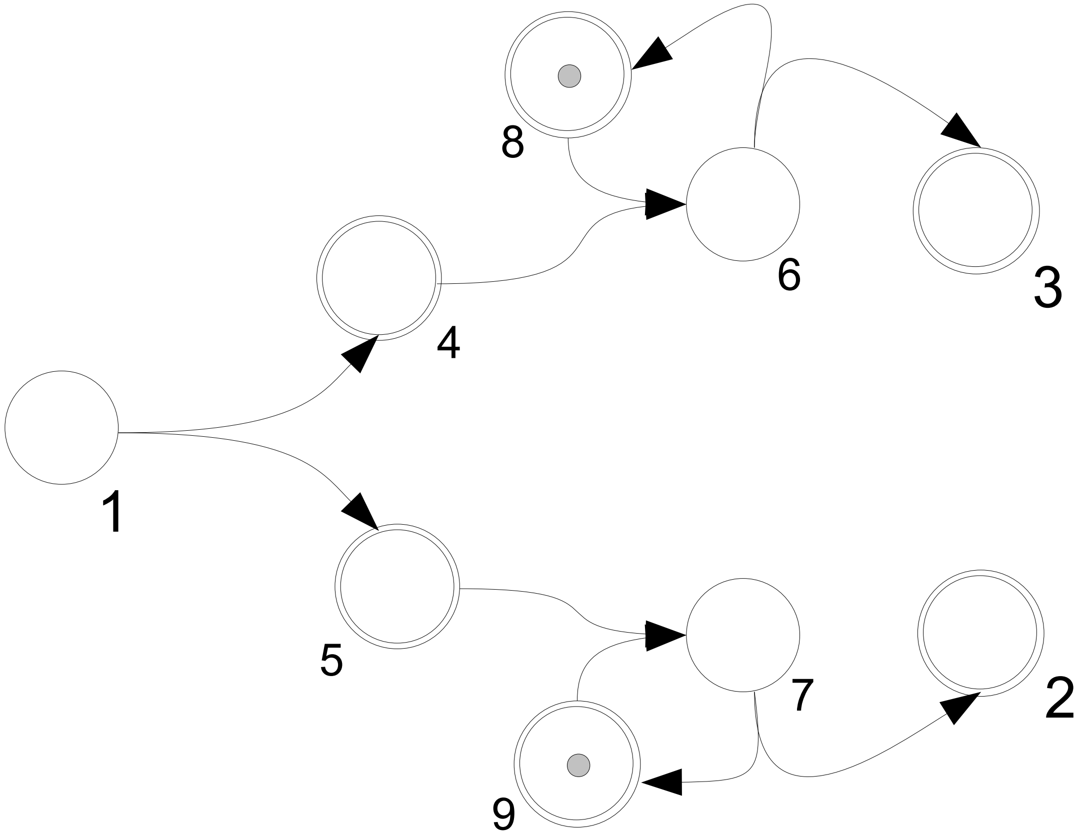
$(5) : (\bullet, 4)(\bullet, 5') \to (\bullet, \#)(\bullet, \#)$ $(6) : (\bullet, 4')(\bullet, 5) \to (\bullet, \#)(\bullet, \#)$

$(7) : (\bullet, 5)(\bullet, 3) \to (\bullet, 4)(\bullet, 4)$ $(8) : (\bullet, 5')(\bullet, 2) \to (\bullet, 4')(\bullet, 4')$

The rules corresponding to the trap cell are the following:

$(\#_1) : (\bullet, \#)(\bullet, 0) \to (\bullet, \bar{\#})(\bullet, \bar{\#})$ $(\#_2) : (\bullet, \bar{\#})(\bullet, 0) \to (\bullet, \#)(\bullet, \#)$

Combination of several pseudo-split and join elements to obtain a split element



The third phase of the construction

The application of a

parallel-shift rule $(t) : (\bullet, i)(\bullet, j) \to (\bullet, k)(\bullet, l)$,

can be simulated by the application of a

join rule $(t') : (\bullet, i)(\bullet, j) \to (\bullet, c_t)(\bullet, c_t)$ **followed** by a

split rule $(t'') : (\bullet, c_t)(\bullet, c_t) \to (\bullet, k)(\bullet, l)$,

where c_t is a new cell introduced to rule $(t) : (\bullet, i)(\bullet, j) \to (\bullet, k)(\bullet, l)$.

Results continued...

$\boxed{\text{GCPSMI with one symbol} + \textbf{presence-move} = \mathbb{N}RE}$

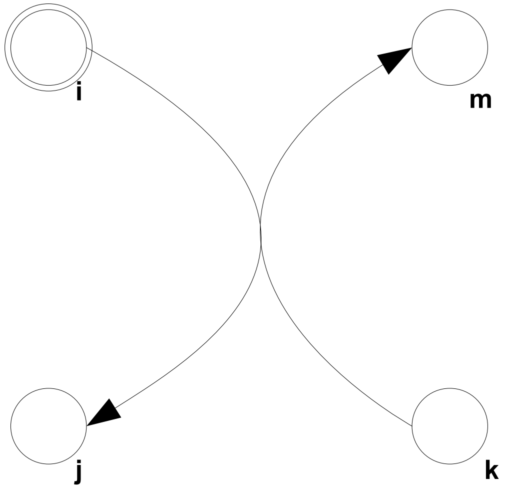
We implement the three **primitive blocks**: the uniport block, the main block, and the zero block.

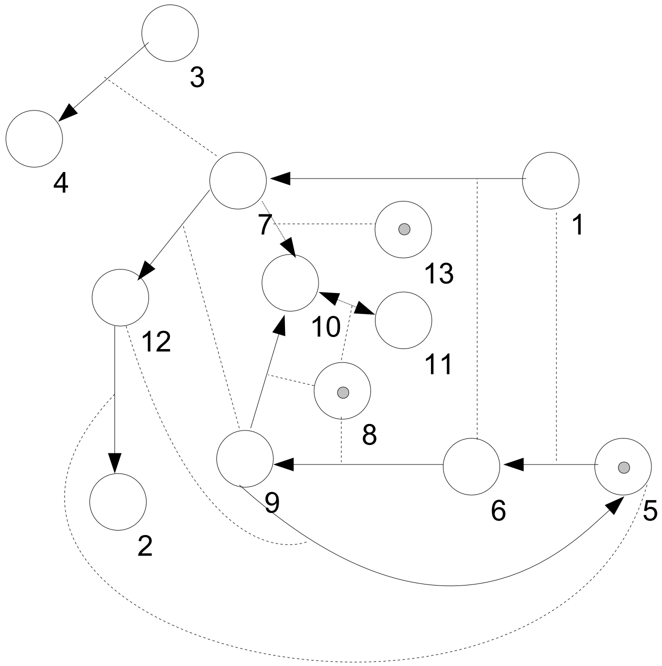
The uniport block

This action is simulated by the following presence-move rule (we suppose that a token is present initially in cell $1'$):

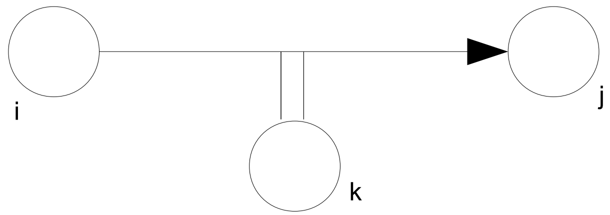
$$(\bullet, 1')(\bullet, 1) \to (\bullet, 1')(\bullet, 2)$$

The implementation of the main block:



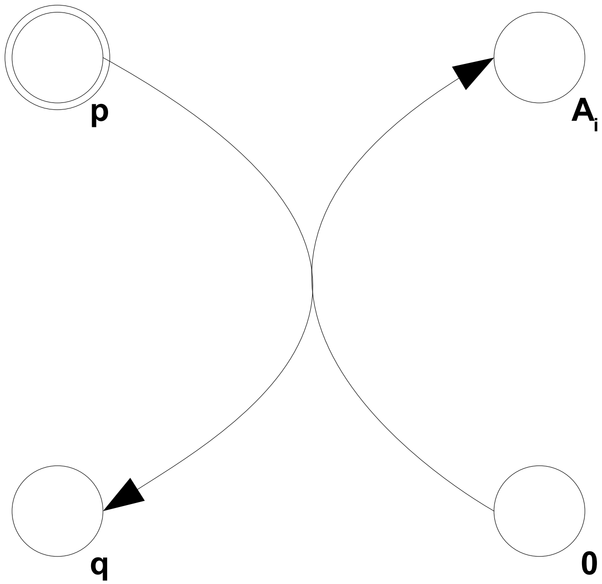


The zero block



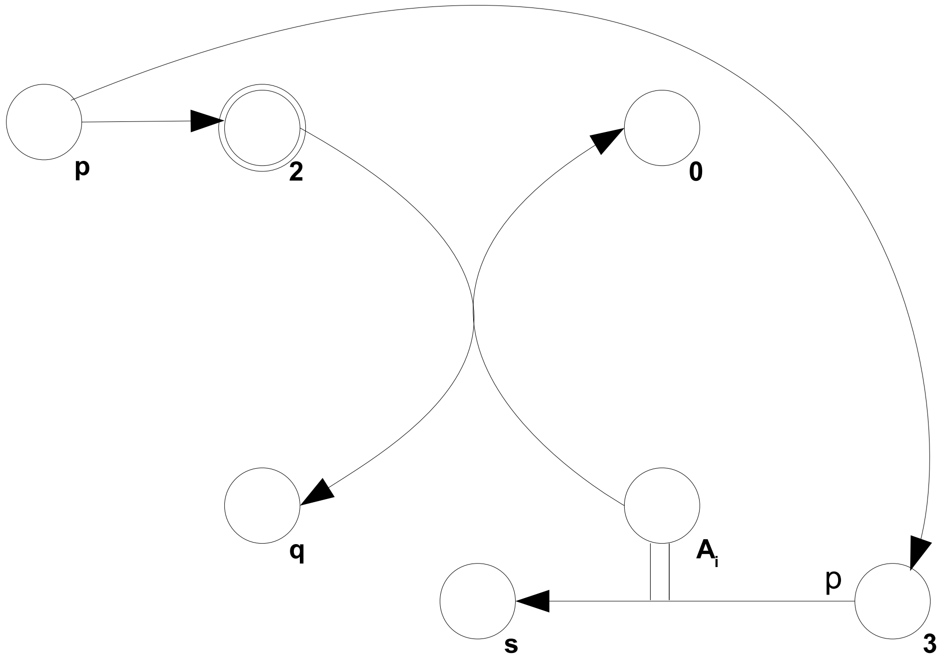
The simulating rules can be obtained from the rules simulating main block with some slight modifications.

The final assembly



Increment: $(p, A+, q)$,

The final assembly



Decrement: $(p, A-, q, s)$

Results continued...

$\boxed{\text{GCPSMI with one symbol} + \textbf{chain} = \mathbb{N}RE}$

With one symbol chain

$$(\bullet, i)(\bullet, j) \rightarrow (\bullet, k)(\bullet, i)$$

is the same as presence-move

$$(\bullet, i)(\bullet, j) \rightarrow (\bullet, i)(\bullet, k)$$

Results continued...

For any GCPSMI Π with **conditional-uniport-in** rules either $N(\Pi)$ is finite or there is a $K \in \mathbb{N}$ such that $l \in N(\Pi)$ for every $l \geq K$.

Proof Idea

1. **Alive path**: a path from the environment to the output cell in the "communication graph" of the initial configuration.

2. When **no such path exists**, $N(\Pi)$ is finite.

3. In the other case: **it is possible to bring any number of tokens from the environment to the output cell**, and moreover

4. **any alive path can be broken**, hence the process of bringing in tokens can be stopped.

Conclusion

- Generalized communicating P systems with one type of rules and one symbol
- Six of the nine rule types makes sense in systems with one symbol
- Systems with five of them are computationally complete
  - direct register machine simulation
  - implementation of the primitive blocks
- Systems with conditional-uniport-in rules generate finite languages or all numbers $l > K$ for some $K \in \mathbb{N}$