Inhibiting the parallelism of P systems

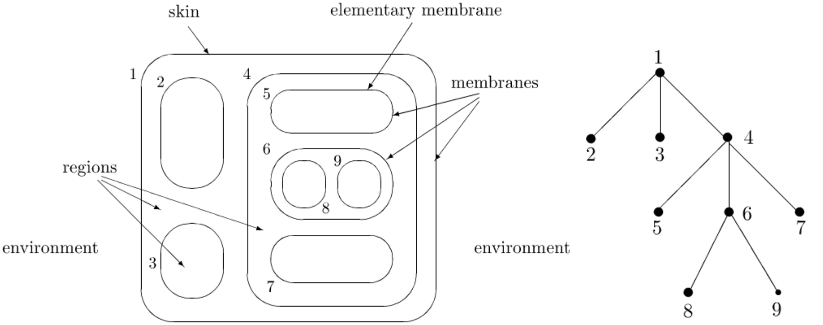
Michal Kováč

FMFI UK

24.6.2014

1. Overview of P systems
    - P systems
    - Variants
2. Sequential P systems with inhibitors
    - Accepting case
    - Generating case

Membrane structure



Contents of the membrane

- multiset of objects
  - $a \mid b \mid b$
- rewriting rules
  - $a \mid b \mid b \to a \mid a_{out} \mid b_{in_6}$
  - $b \to a \mid \delta$

P system

We define a P system as
$\Pi = (V, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m)$, where:

- V is an alphabet of objects
- μ is a membrane structure
- $w_1, w_2, \ldots w_m$ are initial multisets of objects in membranes $1 \ldots m$, $w_i \subseteq \mathbb{N}^V$
- $R_1, R_2, \ldots, R_m$ are sets of rewriting rules in membranes $1 \ldots m$, where

$$R_i \subseteq (\mathbb{N}^V \setminus 0^V) \times \mathbb{N}^{V \times (\{here, out\} \cup \{in_1, \ldots in_m\})}$$

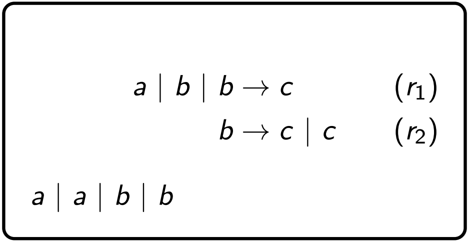
.

Configuration and computational step

- configuration = membrane structure + contents
- computational step: maximal parallelism

Configuration and computational step

- configuration = membrane structure + contents
- computational step: maximal parallelism



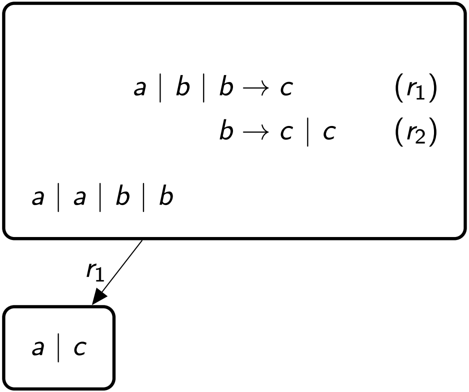
$$a \mid b \mid b \rightarrow c \qquad (r_1)$$

$$b \rightarrow c \mid c \qquad (r_2)$$

$$a \mid a \mid b \mid b$$

Configuration and computational step

- configuration = membrane structure + contents
- computational step: maximal parallelism



$$a \mid b \mid b \rightarrow c \qquad (r_1)$$

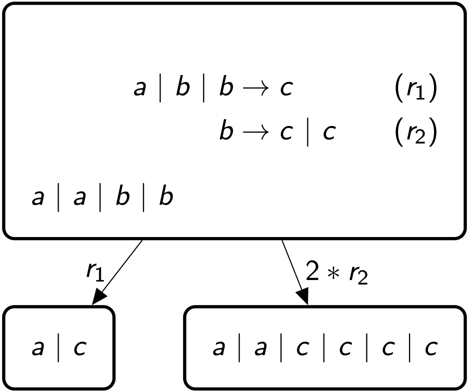
$$b \rightarrow c \mid c \qquad (r_2)$$

$$a \mid a \mid b \mid b$$

$$a \mid c$$

Configuration and computational step

- configuration = membrane structure + contents
- computational step: maximal parallelism



$$a \mid b \mid b \rightarrow c \qquad (r_1)$$
$$b \rightarrow c \mid c \qquad (r_2)$$

$a \mid a \mid b \mid b$

r_1

$2 * r_2$

$a \mid c$

$a \mid a \mid c \mid c \mid c \mid c$

Language

- result of the computation is a multiset of objects, which:
  - have passed the outer membrane
  - is present in a specific membrane at the end

Language

- result of the computation is a multiset of objects, which:
    - have passed the outer membrane
    - is present in a specific membrane at the end
- generating vs. accepting mode

Language

- result of the computation is a multiset of objects, which:
  - have passed the outer membrane
  - is present in a specific membrane at the end
- generating vs. accepting mode
- Parikh mapping: PsRE

Variants of rules

- context (PsRE)

Variants of rules

- context (PsRE)
- cooperative (PsRE) [Păun, 1998]

Variants of rules

- context (PsRE)
- cooperative (PsRE) [Păun, 1998]
- catalytic
  - with 2 catalysts (PsRE) [Freund et al., 2005]
  - with 1 catalyst (open problem)
  - with 1 catalyst and inhibitors (PsRE) [Ionescu and Sburlan, 2004]

Variants of rules

- context (PsRE)
- cooperative (PsRE) [Păun, 1998]
- catalytic
    - with 2 catalysts (PsRE) [Freund et al., 2005]
    - with 1 catalyst (open problem)
    - with 1 catalyst and inhibitors (PsRE) [Ionescu and Sburlan, 2004]
- context-free (PsCF) [Sburlan, 2005]

Variants of rules

- context (PsRE)
- cooperative (PsRE) [Păun, 1998]
- catalytic
  - with 2 catalysts (PsRE) [Freund et al., 2005]
  - with 1 catalyst (open problem)
  - with 1 catalyst and inhibitors (PsRE) [Ionescu and Sburlan, 2004]
- context-free (PsCF) [Sburlan, 2005]
- context-free with inhibitors (PsET0L) [Ionescu and Sburlan, 2004]

Variants of computational step

- maximal parallelism (PsRE)

Variants of computational step

- maximal parallelism (PsRE)
- sequential (equals to VASS, [Ibarra et al., 2005])

Variants of computational step

- maximal parallelism (PsRE)
- sequential (equals to VASS, [Ibarra et al., 2005])
- asynchronous ($\sim$ sequential in most cases) [Freund, 2005]

Variants of computational step

- maximal parallelism (PsRE)
- sequential (equals to VASS, [Ibarra et al., 2005])
- asynchronous ($\sim$ sequential in most cases) [Freund, 2005]
- minimal parallelism (PsRE) [Ciobanu et al., 2007]

Extensions of sequential P systems

- priorities [Ibarra et al., 2005]
- unbounded membrane creation [Ibarra et al., 2005]

Extensions of sequential P systems

- priorities [Ibarra et al., 2005]
- unbounded membrane creation [Ibarra et al., 2005]
- **inhibitors [Kováč, 2014, submitted]**

Extensions of sequential P systems

- priorities [Ibarra et al., 2005]
- unbounded membrane creation [Ibarra et al., 2005]
- **inhibitors [Kováč, 2014, submitted]**
- further study (rules with emptyness detection, . . .)

Register machine

Minsky register machine is $M = (n, P, i, h)$, where:

- n is the number of registers
- P is a set of labeled instructions of type:
  - $(add(r), k, l)$
  - $(sub(r), k, l)$
  - $halt$
- i is the initial instruction
- h is the final instruction

Simulation of register machine

- Contents of register j is represented by the multiplicity of the object a_j
- For an instruction $(add(r), k, l)$ there is a rule $e \to a_j|f$
- For an instruction $(sub(r), k, l)$ there are rules
  - $e|a_j \to f$
  - $e \to z|\not{a}_j$
- Halting rules
  - $h|a_j \to h|\#$ for all $a \leq j \leq n$
  - $\# \to \#$

Overview of the simulation for the generating case

- Simulation of a maximal parallel step
- Phases of membranes: *RUN* and *SYNCHRONIZE*, represented by objects

Major difficulties

- Preventing the rule application on already rewritten objects in the same maximal parallel step

Major difficulties

- Preventing the rule application on already rewritten objects in the same maximal parallel step
    - replace objects on the right side a with a'
    - add *RESTORE* phase

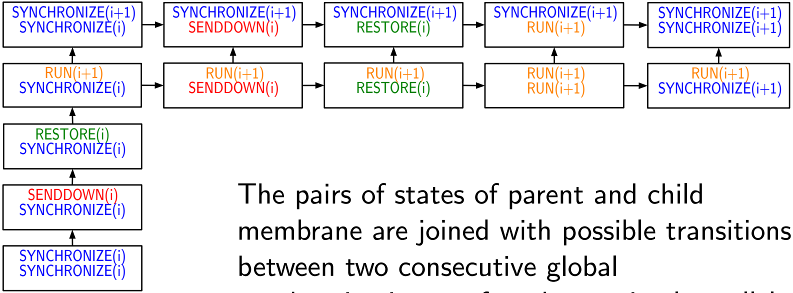
Major difficulties

- Preventing the rule application on already rewritten objects in the same maximal parallel step
  - replace objects on the right side a with a'
  - add *RESTORE* phase
- Sending objects via membranes

Major difficulties

- Preventing the rule application on already rewritten objects in the same maximal parallel step
    - replace objects on the right side a with a'
    - add *RESTORE* phase
- Sending objects via membranes
    - add *SENDDOWN* phase

Parent and child membrane phases



SYNCHRONIZE(i+1)
SYNCHRONIZE(i)

SYNCHRONIZE(i+1)
SENDDOWN(i)

SYNCHRONIZE(i+1)
RESTORE(i)

SYNCHRONIZE(i+1)
RUN(i+1)

SYNCHRONIZE(i+1)
SYNCHRONIZE(i+1)

RUN(i+1)
SYNCHRONIZE(i)

RUN(i+1)
SENDDOWN(i)

RUN(i+1)
RESTORE(i)

RUN(i+1)
RUN(i+1)

RUN(i+1)
SYNCHRONIZE(i+1)

RESTORE(i)
SYNCHRONIZE(i)

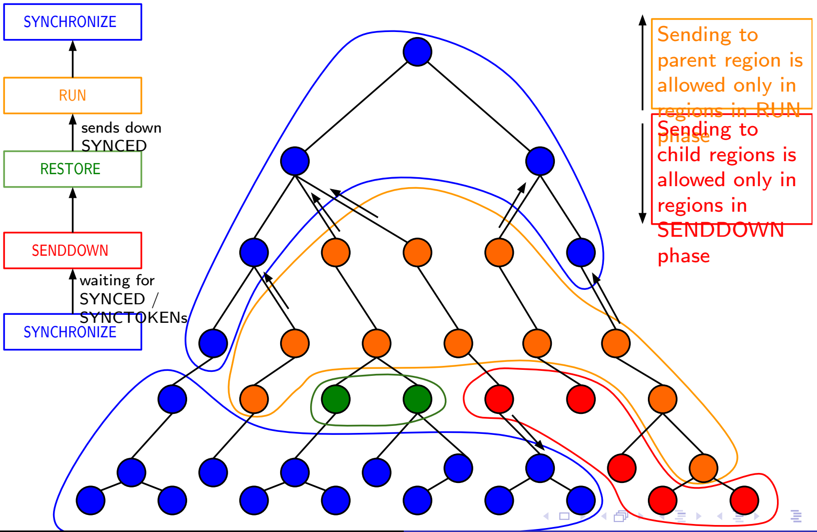
SENDDOWN(i)
SYNCHRONIZE(i)

SYNCHRONIZE(i)
SYNCHRONIZE(i)

The pairs of states of parent and child membrane are joined with possible transitions between two consecutive global synchronizations - after the maximal parallel steps i and i+1

Obr. : Possible pairs of states of parent and child membrane

Snapshot of all membrane states



Sending to parent region is allowed only in regions in RUN phase

Sending to child regions is allowed only in regions in SENDDOWN phase

Thanks for your attention