Adaptive P Systems

Bogdan Aman   Gabriel Ciobanu

Speaker: **Eneia Todoran**
(with thanks and gratitude of the authors)

"A.I.Cuza" University of Iași, România
Romanian Academy, Institute of Computer Science
bogdan.aman@gmail.com, gabriel@info.uaic.ro

4th September 2018

Plan

Introduction

Membrane systems

- are a model of distributed, parallel and non-deterministic systems inspired by cell biology;
- several variants inspired by different aspects of living cells:
  - symport and antiport communication through membranes;
  - catalytic objects;
  - promoters and inhibitors;
  - membrane charges;
  - etc.

  are presented in the Handbook.

Motivation

- In most of the biological systems:
  - exists a delicate dynamic balance depending on the context;
  - different evolution paths are taken depending on the existence or absence of certain substances.
- Inspired by these we define a new class of membrane systems called adaptive P systems.
- These systems use guards to express the dependence to the context.
- Such systems are:
  - able to dynamically adapt to a changing environment;
  - increase the resistance to a variety of failures;
  - increase the expressive power.

Plan

Definition

An adaptive P system of degree d is a tuple

$$\Pi = (O, H, \mu, w_1, \ldots, w_d, R_1, \ldots, R_d, i_0), \text{ where:}$$

- O is a finite non-empty alphabet of objects;
- H is a finite set of labels for membranes;
- μ is a membrane structure with membranes labelled by elements of H;
- $w_1, \ldots, w_d \in O^*$ describe the initial multisets of objects placed in μ;
- i_0 represents the output membrane where the result of the computation is placed; if $i_0 = e$, then the answer is in the surrounding environment;
- R_i $(1 \leq i \leq d)$ is a finite set of rules.

Definition (cont.)

(a) rewriting and communication rules: $u \rightarrow \{g\}v; w$.

(b) division rules: $[u]_h \rightarrow \{g\}[v_1]_h[v_2]_h; [w]_h$.

- The guards are inspired by promoters, inhibitors and kernel P systems.
- A rule $r : u \rightarrow \{g\}v; w$ is applicable to a multiset x if $u \leq x$. Note that the applicability does not depend on the guard.
- A guard is satisfied if $g \leq x - u$, and thus u is rewritten to v; otherwise, u is rewritten to w.
- This means that a guard is used to promote one of the branches and inhibit the other at the same time.

- At each step, in a non-deterministic maximally parallel manner, a multiset of applicable rules is chosen such that no further rule can be added to this multiset.
- Then the application is sequential:
  1. the inner objects, not from guards, evolve in a maximal parallel-manner;
  2. if possible, the objects from guards evolve;
  3. the result is duplicated whenever the surrounding membrane is divided.
- A step can be seen as a macro-step consisting of several micro-steps.
- A micro-step is the application of a rule inside a membrane.
- A macro-step consists in applying micro-steps in parallel in all the membranes, as long as a rule can be applied somewhere.
- A macro-step ends when no further rule is applicable.

Example (Logical AND operation)

We define the following adaptive P system:

$$\Pi = (\{0,1\}, \{1\}, [\ ]_1, w_1, R_1, e), \text{ where}$$

- $R_1 = \{r_1 : 0 \to \{0\}\varepsilon; (0, out), \quad r_2 : 1 \to \{1\}(1, out); \varepsilon\}$

If $w_1 = 0^2$ the rule r_1 can be applied twice. Due to the fact that the guard will be rewritten, the rules are applied sequentially. Namely:

$$0^2 \xrightarrow{r_1} 0 \xrightarrow{r_1} (0, out).$$

Since the guard is satisfied if $g \leq x - u$, after applying the rule r_1 once we have $0 \not\leq 0 - 0$, and so the guard is not satisfied and the second branch/option of the rule is executed, namely 0 is sent to the environment. This models the sensitivity to a dynamic context.

If $w_1 = 01$, then we have two ways of producing the same result:

$$01 \xrightarrow{r_1} (0, out)1 \xrightarrow{r_2} (0, out),$$

$$01 \xrightarrow{r_2} 0 \xrightarrow{r_1} (0, out).$$

If $w_1 = 11$, then we have $1^2 \xrightarrow{r_2} (1, out)1 \xrightarrow{r_2} (1, out)$.

Plan

Subset Sum Problem

- Given a finite set $A = \{a_1, \ldots, a_n\}$, a weight function $w : A \to \mathbb{N}$ and a constant $k \in \mathbb{N}$, decide whether or not there exists a non-empty subset B of A such that its weight is equal to k, namely $w(B) = k$.
- The solution presented in this paper consists of the following stages:
  - Generation and evaluation stage: using membrane division, all the possible subsets are generated and evaluated;
  - Checking stage: in each membrane, the system checks whether or not the weight is equal to k;
  - Output stage: the systems sends out the answer to the environment.

Theorem

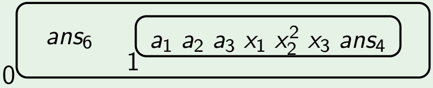
The Subset Sum problem can be solved in linear time by a uniform family of recognizer adaptive P systems, namely an adaptive P system with input that sends the result to the environment.

Example

Consider the Subset Sum problem with $A = \{a_1, a_2, a_3\}$, $k = 3$, $w(a_1) = 1$, $w(a_2) = 2$ and $w(a_3) = 1$. In this case, $n = 3$, $k = 3$, and the initial configuration of the system after adding the input is

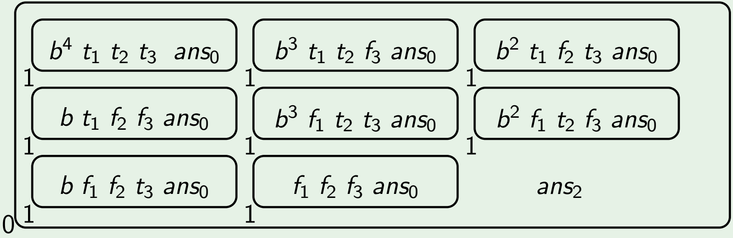
$$[[a_1\ a_2\ a_3\ x_1\ x_2^2\ x_3\ ans_4]_1 ans_6]_0.$$

Graphically, this is illustrated as



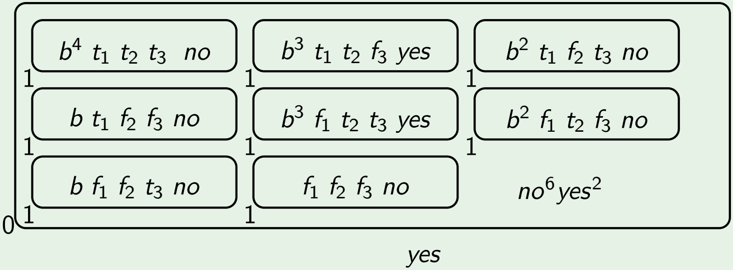
Example (cont.)

- The working space is generated in $n = 3$ steps, and the evaluation stage in 1 step, leading from the initial configuration to the configuration 4 (thus the initial ans_4 object placed in membrane 1 with subscript equal to $n + 1 = 4$):



Example (cont.)

- In the next two stages (checking and output), all ans_0 objects placed in the membranes labelled by 1 are replaced to either *yes* or *no*, depending on the number of b objects existent in membranes 1 (thus the initial ans_6 object placed in membrane 0 with subscript equal to $n+1+2=6$):.
- A copy of each such object is sent to the surrounding membrane 0. In the final step, the remaining ans_0 object placed in the membrane labelled by 0 is replaced by *yes* and sent to the environment.



Plan

Notations

- For a family of languages FL, the family of Parikh images of languages in FL is denoted by $PsFL$.
- The family of recursively enumerable string languages is denoted by RE.
- The set of (Parikh) vectors of non-negative integers accepted by halting computations in Π is denoted by $Ps_{acc}(\Pi)$.
- The families of sets $Ps_{acc}(\Pi)$ computed by adaptive P systems (with guards) with at most m membranes is denoted by $Ps_{acc}OP_m(guard)$.

Theorem

For any $m \geq 1$, $Ps_{acc}OP_m(guard) = PsRE$.

- The theorem illustrates the computational universality (in their accepting variants) of adaptive P systems by simulating a register machine.

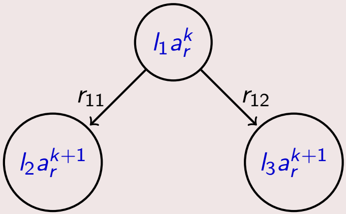
- Instruction $l_1 : (ADD(r), l_2, l_3)$ increases the value of register r by one, followed by a non-deterministic jump to instruction l_2 or l_3.

- $l_1 : (ADD(r), l_2, l_3)$ is simulated by the rules

$$r_{11} : l_1 \to \{a_r\}a_r l_2; a_r l_2$$

$$r_{12} : l_1 \to \{a_r\}a_r l_3; a_r l_3.$$

Any of the rules r_{11} and r_{12} is applied non-deterministically.



Computational Universality

- Instruction $l_1 : (SUB(r), l_2, l_3)$ jumps to instruction l_3 if the register is empty (zero test); otherwise, the value of register r is decreased by one, followed by a jump to instruction l_2

▶ $l_1 : (SUB(r), l_2, l_3)$ is simulated by the rules

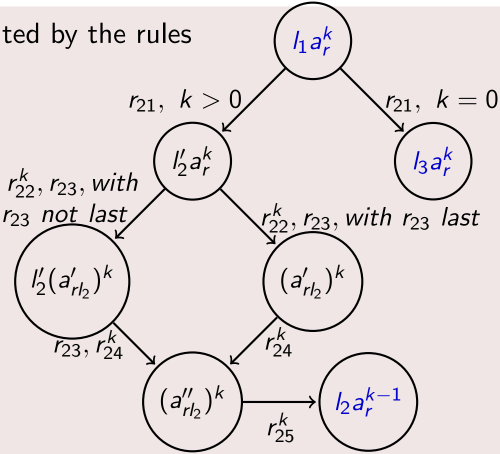
$r_{21} : l_1 \to \{a_r\} l_2'; l_3$

$r_{22} : a_r \to \{l_2'\} a'_{rl_2}; a_r$

$r_{23} : l_2' \to \{a_r\} l_2'; \varepsilon$

$r_{24} : a'_{rl_2} \to \{\varepsilon\} a''_{rl_2}; a''_{rl_2}$

$r_{25} : a''_{rl_2} \to \{a''_{rl_2}\} a_r; l_2$



Conclusion

- We introduced and studied the adaptive P systems, systems able to adjust dynamically their behaviours to the changes in the membrane.
- This approach is inspired by the biological sensitivity to context.
- The main ingredient is a guard used on the right side of the rules; these guards allow to define an adaptive behaviour in the evolution of these new class.
- The generality of the guards, their power and flexibility make the new class of adaptive P systems useful in modelling various biological systems and simulating them in order to solve hard problems in simpler ways.

Many thanks to the speaker!

Please send us your questions by email:

bogdan.aman@gmail.com, gabriel@info.uaic.ro

Thank you!