VARIANTS OF ENERGY-CONTROLLED P SYSTEMS

Artiom Alhazov[1]

Rudolf Freund[2]

Sergiu Ivanov[3]

[1]Institute of Mathematics and Computer Science, Moldova

[2]TU Wien, Austria

[3]Université Paris Est, France

NIT 2016, Valencia

Overview

P Systems

P systems are formal systems processing multisets of objects in a cell-like membrane structure.
In the basic model introduced by Gheorghe Păun, in each transition step, a maximal multiset of rules is applied to the objects in each membrane region.

- Gh. Păun:
  Computing with membranes.
  J. Comput. Syst. Sci., 61 (2000), 108–143 (see also TUCS Report 208, November 1998, `www.tucs.fi`).
- Gh. Păun, G. Rozenberg, A.Salomaa (Eds.):
  The Oxford Handbook of Membrane Computing.
  Oxford Univ. Press, 2010.
- The P Systems Website: `http://ppage.psystems.eu`.

The Basic Model of P Systems

A (cell-like) P system is a construct

$\Pi = (O, C, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, f_I, f_O)$ where

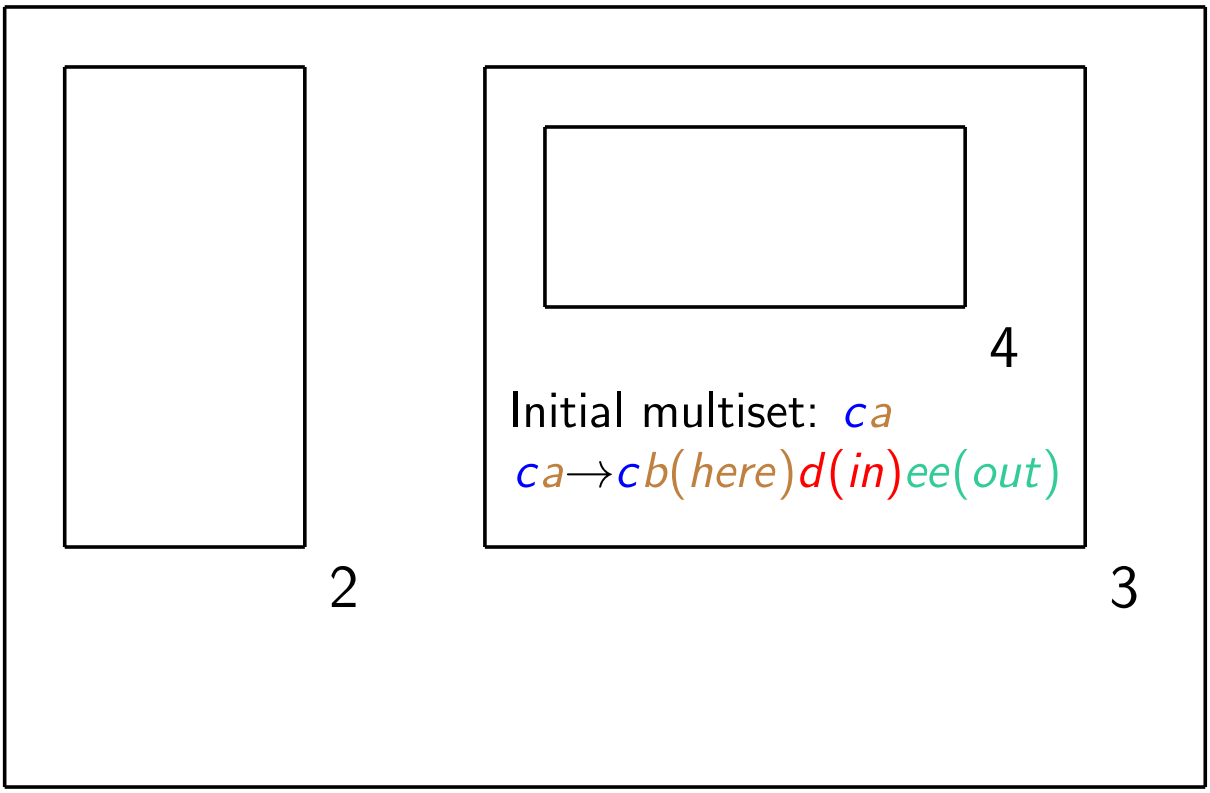
- O is the alphabet of objects,
- $C \subset O$ is the set of catalysts,
- μ is the membrane structure (with m membranes),
- $w_1, \ldots, w_m$ are multisets of objects present in the m regions of μ at the beginning of a computation,
- $R_1, \ldots, R_m$ are finite sets of rules, associated with the membrane regions of μ,

The Basic Model of P Systems

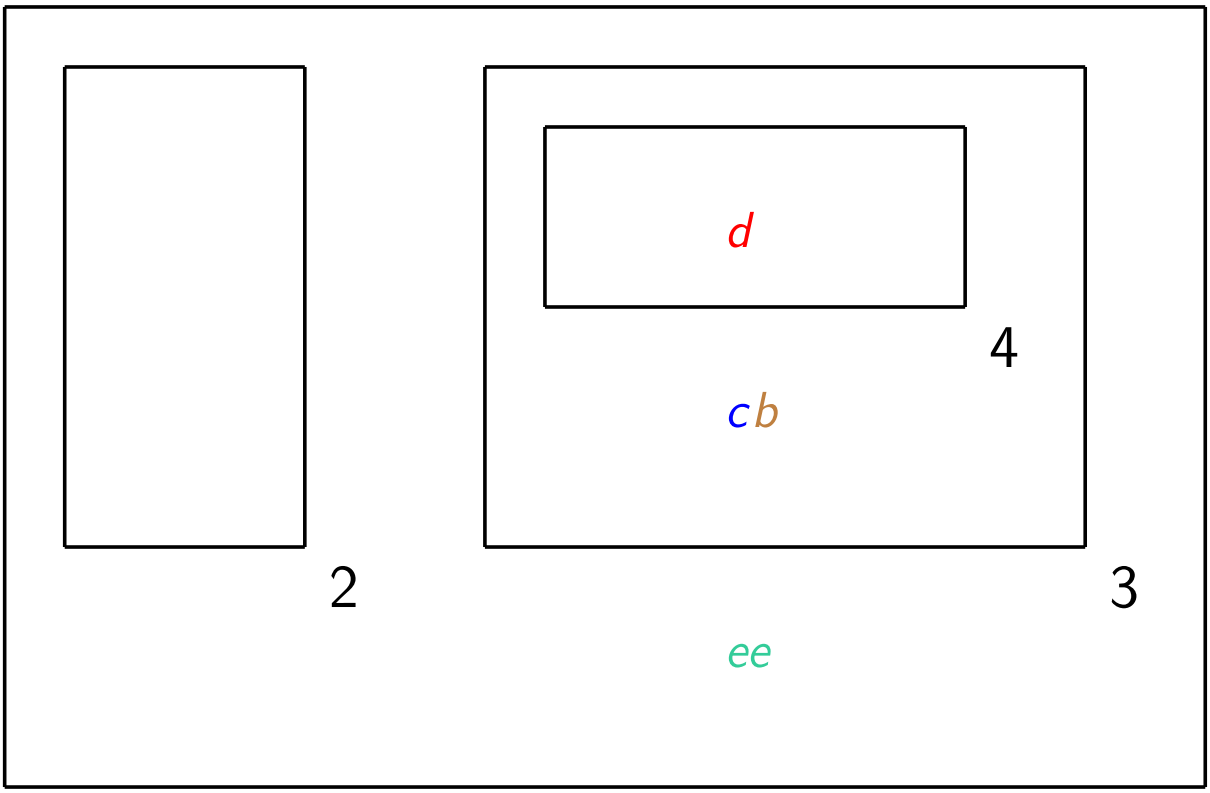
- f_I is the label of the membrane region where the inputs are put at the beginning of a computation (in the accepting/computing case).
- f_O is the label of the membrane region from which the outputs are taken at the end of a halting computation (in the generative/computing case).

$f_I = 0/f_I = 0$ indicates that the output/input is taken from the environment.

Structure of a P System



Structure of a P System



The Rules in the Basic Models of P Systems

A rule $u \to v$ is called

- *cooperative* if $u \geq 2$,
- *non-cooperative* if $u = 1$, and
- *catalytic* if it is of the form $ca \to cv$, where $c \in C$ is a special object which never evolves and never passes through a membrane, it just assists object a to evolve to the multiset v.

Catalytic P system: catalytic rules as well as non-cooperative rules.
Purely catalytic P system: only catalytic rules.

Derivation Modes

- sequential derivation mode (*sequ*):
  exactly one rule is used in each step.
- asynchronous derivation mode (*asyn*):
  an arbitrary number of rules is used in each step.
- maximally parallel derivation mode (*max*):
  in each step, a multiset of rules from the sets $R_1, \ldots, R_m$ of rules is chosen in such a way that no further rule can be added to it so that the obtained multiset would still be applicable to the current configuration.

Other Variants of Maximal Derivation Modes

As the rules compete for the objects present in the current configuration, we also consider the following:

- maximal number of objects derivation mode ($max_{objects}$):
  in each derivation step, from the maximal multisets of rules only those are taken which affect the maximal number of objects.
- maximal number of rules derivation mode (max_{rules}):
  in each derivation step, from the maximal multisets of rules only those are taken which use the maximal number of rules.

Set Derivation Modes

The derivation modes where we consider sets of rules, i.e., each rule can be used at most once in each step, are called set derivation modes.

- asynchronous set derivation mode (*sasyn*): in each derivation step, one applicable set of rules is used.
- set maximal derivation mode (*smax*): in each step, a set of rules from the sets $R_1, \ldots, R_m$ is chosen in such a way that no further rule can be added to it so that the obtained set of would still be applicable to the current configuration.

Set Derivation Modes

- $smax_{objects}$:
  in each derivation step, from the maximal sets of rules only those are taken which affect the maximal number of objects.
- $smax_{rules}$:
  in each derivation step, from the maximal sets of rules only those are taken which use the maximal number of rules.

Computations in a P System

The membranes and the objects present in the compartments of a system at a given time form a *configuration*.

We start from the given *initial configuration* and using the rules as explained above, we get *transitions* among configurations.

A sequence of transitions forms a *computation*.

Halting in P Systems

A computation is called *halting* if it reaches a configuration where no rule can be applied.

Halting with states means that the computation reaches a configuration which fulfills a specific (computable) condition.

Adult halting means that the P system reaches a configuration which does not change any more with the application of any (multi)set of rules.

Results of Halting Computations in P Systems

In generating P systems, with a halting computation we associate a *result*, in the form of the number of (different) objects present in region f_O in the halting configuration.

In accepting P systems, the input given in a membrane region $f_I \neq 0$, is accepted by a halting computation.

Register Machines – a *Computationally Complete* Model of Devices Computing with Numbers

A *register machine* is a tuple

$$M = (d, B, l_0, l_h, R) \quad \text{where}$$

- d is the number of registers,
- R is the set of instructions bijectively labeled by elements of B,
- $l_0 \in B$ is the initial label, and
- $l_h \in B$ is the final label.
  The instructions of M in R can be of the following forms:

Register Machines – Instructions

- $l_1 : (\texttt{ADD}(j), l_2, l_3)$,
  with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq d$.
  Increase the value of register j by one, and non-deterministically jump to instruction l_2 or l_3. This instruction is usually called *increment*.
- $l_1 : (\texttt{SUB}(j), l_2, l_3)$,
  with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq d$.
  If the value of register j is zero then jump to instruction l_3, otherwise decrease the value of register j by one and jump to instruction l_2. The two cases of this instruction are usually called *zero-test* and *decrement*, respectively.
- $l_h : \texttt{HALT}$. Halt the register machine program.

Register Machines – Configurations and Computations

A *configuration* of a register machine is described by the contents of each register and by the value of the current label, which indicates the next instruction to be executed.
Computations start by executing the first instruction of R (labeled with l_0), and terminate with reaching the HALT-instruction.

Register machines provide a computationally complete model for computations with natural numbers.

Two Variants of Energy Control

In symbol energy-controlled P systems, fixed integer values of energy are assigned to each symbol in the system, i.e., instead of O we consider the set O_E consisting of pairs $[x, f(x)]$ with $x \in O$ and $f : O \to \mathbb{Z}$ being a function assigning a unique energy value to each symbol in O.
We extend f in the natural way to multisets over O.
The energy balance of a rule $u \to v$ then is $f(v) - f(u)$.

In rule energy-controlled P systems, the energy is directly assigned to the rules only.

Derivation Modes and Energy Control

All the derivation modes can also be used for symbol energy-controlled P systems and rule energy-controlled P systems.

In addition to the restrictions given by the derivation mode itself, the multisets or sets of rules then also must fulfill the condition of yielding the minimal amount of energy.

Simple P Systems with Cooperative Rules

Simple symbol or rule energy-controlled P systems with cooperative rules have only one membrane (the skin membrane), which also serves as input and output membrane, and cooperative rules of the form $u \rightarrow v$; $|uv|$ is called its *size*.

$\Pi = (O_E, w_1, R_1)$ where

- O_E is the alphabet of objects with unique integer energy values,
- w_1 is the finite multiset of objects over O_E present in the skin membrane at the beginning of a computation,
- R_1 is a finite set of cooperative rules over O_E.

Simple Symbol-Controlled P Systems with Cooperative Rules

Theorem

For any register machine $M = (d, B, l_0, l_h, R)$, *with* $m \leq d$ *being the number of decrementable registers, we can construct a simple symbol energy-controlled P system with cooperative rules of size* ≤ 3 $\Pi = (O, w_1, R_1)$ *working in any of the derivation modes* $sequ$, $asyn$, $sasyn$, max, $smax$, max_{rules}, $max_{objects}$, $smax_{rules}$, $smax_{objects}$ *and simulating the computations of* M *such that*

$$|R_1| \leq |ADD^1(R)| + 2 \times |ADD^2(R)| + 2 \times |SUB(R)| + 1.$$

Simple Symbol-Controlled P Systems with Cooperative Rules

Proof. Let $M = (m, B, l_0, l_h, R)$ be an arbitrary register machine. We now construct a simple symbol energy-controlled P system with cooperative rules of size 3 simulating M in real time.
The number in register r is represented by the corresponding number of symbol objects $[o_r, 1]$.
We also assume all objects in B to have energy value 1.
The number in between the brackets $\langle$ and $\rangle$ describes the total amount of energy consumed by the corresponding rule.

Simple Symbol-Controlled P Systems with Cooperative Rules

A deterministic ADD-instruction $p : (ADD(r), q)$ is simulated by the rule

$[p, 1] \rightarrow [o_r, 1]\,[q, 1]\,\langle 1 \rangle$.

An ADD-instruction $p : (ADD(r), q, s)$ is simulated by the two rules

$[p, 1] \rightarrow [o_r, 1]\,[q, 1]\,\langle 1 \rangle$ and

$[p, 1] \rightarrow [o_r, 1]\,[s, 1]\,\langle 1 \rangle$.

Simple Symbol-Controlled P Systems with Cooperative Rules

A SUB-instruction $p : (SUB(r), q, s)$ is simulated by the rules

$[p, 1]\,[o_r, 1] \to [q, 1]\,\langle -1 \rangle$ and

$[p, 1] \to [s, 1]\,\langle 0 \rangle$.

As the total energy balance of the rule $[p, 1]\,[o_r, 1] \to [q, 1]$ is -1, in case the register is not empty, it has priority over the rule $[p, 1] \to [s, 1]$, which has the total energy balance 0 and performs the zero-test case.

Simple Symbol-Controlled P Systems with Cooperative Rules

For the final label l_h, we take the rule $[l_h, 1] \to [\lambda, 0] \langle -1 \rangle$.

In the case of a deterministic register machine, the simulation by the P system is deterministic, too.

We also observe that the construction works for every derivation mode. □

Simple Rule-Controlled P Systems with Cooperative Rules

Corollary

For any register machine $M = (d, B, l_0, l_h, R)$, *with* $m \leq d$ *being the number of decrementable registers, we can construct a simple rule energy-controlled P system with cooperative rules of size* ≤ 3 $\Pi = (O, w_1, R_1)$ *working in any of the derivation modes* $sequ, asyn, sasyn, max, smax, max, max_{rules}, max_{objects}, smax, smax_{rules}, smax_{objects}$ *and simulating the computations of* M *such that*

$$|R_1| \leq |ADD^1(R)| + 2 \times |ADD^2(R)| + 2 \times |SUB(R)| + 1.$$

Simple Rule-Controlled P Systems with Cooperative Rules

Proof.

We can immediately take over the proof of the preceding theorem by just omitting the energy values assigned to the objects and taking the values given between the brackets $\langle$ and $\rangle$ as the energy values assigned to the corresponding rules.

□

(Purely) Catalytic Energy-Controlled Simple P Systems

Theorem

For any register machine $M = (d, B, l_0, l_h, R)$, with $m \leq d$ being the number of decrementable registers, we can construct a symbol or rule energy-controlled simple catalytic or purely catalytic P system

$$\Pi = (O_E, C_E, w_1 = [l_0, 1]\,[c_0, 1] \ldots [c_m, 1]\,, R_1)$$

working in any of the maximal derivation modes max, $smax$, max_{rules}, $smax_{rules}$, $max_{objects}$, $smax_{objects}$ *and simulating the computations of M such that*

$$|R_1| \leq |ADD^1(R)| + 2 \times |ADD^2(R)| + 4 \times |SUB(R)| + 2 \times m + 1.$$

(Purely) Catalytic Energy-Controlled Simple P Systems

Proof. Let $M = (m, B, l_0, l_h, R)$ be an arbitrary register machine. Again the number in register r is represented by the corresponding number of symbol objects $[o_r, 1]$.
For each decrementable register r, $1 \leq r \leq m$, we use one catalyst $[c_r, 1]$, and the catalyst $[c_0, 1]$ is used for the program symbols.

$$D_m = \prod_{i \in [1..m]} [d_i, 0] ,$$

$$D_{m,r} = \prod_{i \in [1..m] \setminus \{r\}} [d_i, 0] .$$

(Purely) Catalytic Energy-Controlled Simple P Systems

A deterministic ADD-instruction $p : (ADD(r), q)$ is simulated by the rule

$[c_0, 1]\,[p, 1] \to [c_0, 1]\,[o_r, 1]\,[q, 1]\,D_m\ \langle 1 \rangle.$

An ADD-instruction $p : (ADD(r), q, s)$ is simulated by the two rules

$[c_0, 1]\,[p, 1] \to [c_0, 1]\,[o_r, 1]\,[q, 1]\,D_m\ \langle 1 \rangle$

$[c_0, 1]\,[p, 1] \to [c_0, 1]\,[o_r, 1]\,[s, 1]\,D_m\ \langle 1 \rangle.$

(Purely) Catalytic Energy-Controlled Simple P Systems

A SUB-instruction $p : (SUB(r), q, s)$ is simulated by the following rules:

1. $[c_0, 1]\,[p, 1] \to [c_0, 1]\,[\bar{p}, 1]\,D_{m,r}\ \langle 0 \rangle$;
2. $[c_r, 1]\,[o_r, 1] \to [c_r, 1]\,[e, -3]\,D_m\ \langle -4 \rangle$,
   $[c_0, 1]\,[\bar{p}, 1] \to [c_0, 1]\,[\hat{p}, 2]\ \langle 1 \rangle$,
   $[c_r, 1]\,[\bar{p}, 1] \to [c_r, 1]\,[s, 1]\,D_m\ \langle 0 \rangle$; if there exists at least one register symbol o_r, then the first two rules yield the energy balance -3 and thus supersede the third rule, which taken alone (and leaving c_0 idle) supersedes the second rule if it cannot be combined with the first one.

(Purely) Catalytic Energy-Controlled Simple P Systems

Moreover, the catalysts c_i, $i \neq r$, cannot be used with a register object o_i, as the rule

$$[c_i, 1]\,[d_i, 0] \rightarrow [c_i, 1]\,[e, -3]^2 \ \langle -6 \rangle$$

has a higher negative energy value.

In the decrement case, we finish with a third step using the rule

$[c_0, 1]\,[\hat{p}, 2] \rightarrow [c_0, 1]\,[q, 1]\,D_m \ \langle -1 \rangle$.

We finally observe that, if M is deterministic, then Π works in a deterministic way, too. □

Conclusion

- We have considered several variants of P systems with the multisets or sets of rules chosen according to the derivation mode together with the condition of yielding the minimal total amount of energy.
- The simulations of register machines showing computational completeness can even be carried out in a deterministic way for deterministic register machines.
- Many more variants wait for future research.

Muchas gracias!

Thank you very much!

Danke schön!