P Automata with Controlled Use of Minimal Communication Rules

Rudolf Freund[1], Marian Kogler[1],
and Sergey Verlan[2]

[1] Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria
[2] LACL, Département Informatique
UFR Sciences et Technologie, Université Paris XII
61, av. Général de Gaulle, 94010 Créteil, France

Email: rudi@emcc.at, marian@emcc.at
and verlan@univ-paris12.fr

NCMA 2009

Overview

Introduction to Membrane Systems

P Systems

- introduced by by Gheorghe Păun in 1998,
- inspired by cell functioning,
- multisets of objects evolve in parallel,
- in a hierarchical membrane structure.

PĂUN, Gh., Computing with membranes, J. of Computer and System Sciences 61, 1 (2000), 108–143, and TUCS Research Report 208 (1998) (`http://www.tucs.fi`).

PĂUN, Gh., Membrane Computing. An Introduction, Springer-Verlag, Berlin 2002.

Analyzing P Systems / P Automata

introduced 2002:

- Analyzing P Systems: R. Freund and M. Oswald;
- P Automata: E. Csuhaj-Varjú and Gy. Vaszil;
- act as acceptors rather than as generators;
- multiset is accepted if and only if the system/automaton halts.

CSUHAJ-VARJÚ, E., VASZIL, G., P automata or purely communicating accepting P systems, in: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), Membrane Computing, International Workshop, WMC-CdeA 2002, Curteă de Argeș, Romania, August 19–23, 2002, Revised Papers, Lecture Notes in Computer Science **2597**, Springer, 2003, 219–233.

FREUND, R., OSWALD, M., A short note on analysing P systems with antiport rules, Bulletin of the EATCS 78, 2002, 231–236.

Register Machines

A *deterministic register machine* is a construct $M = (n, B, l_0, l_h, I)$, where

- n is the number of registers,
- B is a set of instruction labels,
- l_0 is the start label,
- l_h is the halt label (assigned to HALT only), and
- I is a set of instructions of the following forms:
  - $l_i : (\mathtt{ADD}(r), l_j)$ add 1 to register r, and then go to the instruction labeled by l_j;
  - $l_i : (\mathtt{SUB}(r), l_j, l_k)$ if register r is non-empty (non-zero), then subtract 1 from it and go to the instruction labeled by l_j, otherwise go to the instruction labeled by l_k;
  - $l_h : \mathtt{HALT}$ the halt instruction.

Register Machines – Computations

A register machine M accepts a set of (vectors of) natural numbers in the following way:

- start with the instruction labeled by l_0, with the first registers containing the input as well as all other registers being empty,
- apply instructions as indicated by the labels and by the contents of the registers,
- accept the input number (vector) if the `HALT` instruction is reached.

It is known that in this way we can accept all recursively enumerable sets of (vectors of) natural numbers.

Communication P Automata

Definition

A *communication P automaton* is a construct

$$\Pi = (O, T, \mu, E, w_0, w_1, ..., w_d, i_0, R) \text{ where}$$

1. O is a finite alphabet of objects;
2. $T \subseteq O$ is the alphabet of terminal objects;
3. μ is a membrane structure of d membranes with labels i, $1 \leq i \leq d$, the skin membrane always has the index 1; the environment is indicated by 0;
4. $E \subseteq O$ is the alphabet of objects occurring infinitely often in the environment;
5. $w_0, w_1, ..., w_d$ initial multisets of the environment (w_0 only contains objects from $O - E$) and membranes i, $1 \leq i \leq d$;
6. i_0, $1 \leq i_0 \leq d$, is the input membrane;
7. R is a set of communication rules.

Communication Rules

Definition

Symport rules in R are of the form $x[_i \rightarrow [_i x$ meaning that the multiset x from outside membrane i is moved into the region inside membrane i, or $[_i x \rightarrow x[_i$ meaning that the multiset x from inside membrane i is moved into the region surrounding membrane i, with $x \in O^+$ and $1 \leq i \leq d$.

For symport rules $x[_1 \rightarrow [_1 x$, at least one symbol from x has to be from $O - E$.

Definition

Antiport rules in R are of the form $x[_i y \rightarrow y[_i x$ with $x, y \in O^+$ and $1 \leq i \leq d$, meaning that the multiset x from outside membrane i is exchanged with the multiset y in the region inside membrane i.

Weight of Communication Rules

Definition

The weight of a symport rule $x[_i \rightarrow [_i x$ or $[_i x \rightarrow x[_i$ is defined as $|x|$, The weight of an antiport rule $x[_i y \rightarrow y[_i x$ is defined as $max(|x|, |y|)$.

Definition

If we consider symport rules of any weight, we write sym_*; if we only consider symport rules with weight $\leq n$, we write sym_n; sym_2 rules are also called *minimal symport* rules; finally, sym_1 rules are called *uniport* rules.

If we consider antiport rules of any weight, we write $anti_*$; if we only consider antiport rules with weights $\leq n$, we write $anti_n$; $anti_1$ rules are also called *minimal antiport* rules.

Computations in Communication P Automata

A *configuration* C of Π is a $(d+1)$-tuple of multisets over O $(u_0, u_1 \ldots, u_d)$; the *initial configuration* of Π, C_0, is described by w_0,w_1, ..., w_d, i.e., $C_0 = (w_0, w_1, \ldots, w_d)$.
The set of all multisets of rules from R *applicable* to C is denoted by $Appl(\Pi, C)$.

To narrow the possible set of multisets of rules that can be applied to a given configuration, we may apply different *transition modes*. For the transition mode ϑ, the selection of multisets of rules applicable to a configuration C is denoted by $Appl(\Pi, C, \vartheta)$.

Transition Modes

Definition

For the *maximally parallel* transition mode (*max*), we define

$$
\begin{aligned}
Appl\,(\Pi, C, max) \quad = \quad & \{R' \mid R' \in Appl\,(\Pi, C) \text{ and there is} \\
& \text{no } R'' \in Appl\,(\Pi, C) \text{ with } R'' \supsetneqq R'\}\,.
\end{aligned}
$$

FREUND, R., VERLAN, S., A formal framework for P systems, in: G. Eleftherakis, P. Kefalas, Gh. Păun (Eds.), Pre-proceedings of Membrane Computing, International Workshop – WMC8, Thessaloniki, Greece, 2007, 317–330.

Minimally Parallel Transition Mode

For the *minimally parallel* mode, we need an additional feature for the set of rules R, i.e., we consider a partitioning of R into (not necessarily disjoint) subsets R_1 to R_h. Usually, this partitioning of R may coincide with a specific assignment of the rules to the membranes.

In an informal way, it can be described as applying multisets such that from every set R_j, $1 \leq j \leq h$, at least one rule – if possible – has to be used:

Definition

For the *minimally parallel* transition mode (min), we define

$$\begin{aligned} Appl\,(\Pi, C, min) \quad = \quad & \{R' \mid R' \in Appl\,(\Pi, C) \text{ and} \\ & \text{there is no } R'' \in Appl\,(\Pi, C) \\ & \text{with } R'' \supsetneq R',\ (R'' - R') \cap R_j \neq \emptyset \\ & \text{and } R' \cap R_j = \emptyset \text{ for some } j,\ 1 \leq j \leq h\}\,. \end{aligned}$$

k-Restricted Minimally Parallel Transition Mode

Definition

For the *k-restricted minimally parallel* transition mode (min_k), we define

$$\begin{aligned} Appl\,(\Pi, C, min_k) &= \{R' \mid R' \in Appl\,(\Pi, C, min) \text{ and} \\ &\quad |R' \cap R_j| \leq k \text{ for all } j,\ 1 \leq j \leq h\}\,. \end{aligned}$$

- CIOBANU, G., PAN, L., PĂUN, Gh., PÉREZ-JIMÉNEZ, M.J., P systems with minimal parallelism, Theoretical Computer Science 378 (1) (2007), 117–130.
- FREUND, R., VERLAN, S., (Tissue) P systems working in the k-restricted minimally parallel derivation mode, in: E. Csuhaj-Varjú, R. Freund, M. Oswald, K. Salomaa (Eds.), Proceedings of the International Workshop on Computing with Biomolecules, Österreichische Computer Gesellschaft, 2008, 43–52.

Communication P Automaton with Rule Control

Definition

A *communication P automaton with rule control* is a construct

$$\Pi' = \left(O, T, \mu, E, w_0, w_1, ..., w_d, i_0, R, R'_1, ..., R'_m, K\right)$$

where

1. $\Pi = (O, T, \mu, E, w_0, w_1, ..., w_d, i_0, R)$ is a communication P automaton with $O, T, \mu, E, w_0, w_1, ..., w_d, i_0$, and R being defined as before;
2. $R'_1, ..., R'_m$ is a partitioning of R into (non-empty, but not necessarily disjoint) subsets;
3. $K \subseteq \{0, 1\}^m$ is a set of control vectors controlling the applicability of multisets of rules from R.

Computations

Definition

For some given transition mode ϑ,

$$\begin{aligned} Appl\,(\Pi', C, \vartheta) \quad = \quad & \{ R' \mid R' \in Appl\,(\Pi, C, \vartheta) \text{ and} \\ & \text{there exists a vector } v \in K \text{ such that} \\ & \text{for all } j \text{ with } 1 \leq j \leq m \text{ it holds that} \\ & v\,(j) = 1 \text{ implies } R' \cap R_j \neq \emptyset \text{ and} \\ & v\,(j) = 0 \text{ implies } R' \cap R_j = \emptyset \,\} \,. \end{aligned}$$

A *computation* in Π' checking for the acceptance of a multiset w consists of a sequence of transitions starting from the initial configuration $C_0' = (w_0', w_1', \ldots, w_d')$ with $w_i' = w_i$ for $0 \leq i \leq m$ and $i \neq i_0$ and $w_i' = w_i + w$ for $i = i_0$. A multiset w is accepted if and only if there exists a halting computation of Π' on C_0'.

Example

Let $\Pi' = (O, T, [_1\ ]_1, E, w_0, w_1, 1, R, R_1, R_2, R_3, R_4, R_5, K)$ be a communication P automaton with rule control with

$$
\begin{aligned}
O &= \{a, b, p_1, p_2\}\,, \\
T &= \{a\}\,, \\
E &= \{b\}\,, \\
w_0 &= \{\}\,, \\
w_1 &= \{p_1 p_2\}\,, \\
R &= R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5; \\
R_1 &= \{[_1 p_1 a \rightarrow p_1 a [_1\}\,, \\
R_2 &= \{[_1 p_2 a \rightarrow p_2 a [_1\}\,, \\
R_3 &= \{p_1 b [_1 \rightarrow [_1 p_1 b\}\,, \\
R_4 &= \{p_2 [_1 \rightarrow [_1 p_2\}\,, \\
R_5 &= \{p_1 [_1 p_2 \rightarrow p_2 [_1 p_1, p_2 [_1 p_1 \rightarrow p_1 [_1 p_2\}\,, \\
K &= \{(1,1,0,0,0)\,, (1,0,0,0,0)\,, (0,0,1,1,0)\,, (0,0,0,0,1)\}\,.
\end{aligned}
$$

Example

In the maximally parallel mode, this automaton starts with the multiset $p_1 p_2 a^n$ for some $n \geq 0$ (where a^n is the input) and first applies the rules in R_1 and R_2 in parallel, thereby fulfilling $(1, 1, 0, 0, 0)$, exporting two symbols a from the skin membrane into the environment; if only one symbol a is available, then only $[_1 p_1 a \rightarrow p_1 a [_1$ is applied thereby fulfilling $(1, 0, 0, 0, 0)$. If both p_1 and p_2 have been moved out, then the rules in R_3 and R_4 are executed at the same time according to $(0, 0, 1, 1, 0)$; these two steps have replaced two symbols a by one symbol b. The P automaton Π' repeats this process until at most one symbol a is left in the skin membrane. If no a is left, i.e., n has been an even number, the automaton terminates with having $p_1 p_2 b^{n/2}$ in the skin membrane. If one a is left, i.e., n has been an odd number, then only R_1 is applicable and the automaton ends up in an infinite loop with $p_1 [_1 p_2 \rightarrow p_2 [_1 p_1$ and $p_2 [_1 p_1 \rightarrow p_1 [_1 p_2$.
The automaton therefore exactly accepts a^{2m} for any $m \geq 0$.

Communication P Automaton with Rule Control

Definition

For some given transition mode ϑ, by

$$NO_lP_dK_m(\vartheta)\,[\text{rule types}]\;\;(PsO_lP_dK_m(\vartheta)\,[\text{rule types}]\,)$$

we define the sets of (vectors of) natural numbers accepted by communication P automata with rule control working in the transition mode ϑ in d membranes using l objects and a partitioning with m rule sets, allowing rules of the types specified in $[\text{rule types}]$; if any of the numbers d, m, l are unbounded, we write $*$ instead.

Computational Completeness

By simulating deterministic register machines, we can show that communication P automata with rule control using only minimal symport rules (sym_2 rules) or minimal antiport rules ($anti_1$ rules) together with uniport rules (sym_1 rules) in only one membrane are computationally complete.

Theorem

For $X \in \{N, Ps\}$,

$$\begin{aligned} XRE &= XO_*P_1K_*(max)[sym_2] \\ & XO_*P_1K_*(max)[anti_1, sym_1]. \end{aligned}$$

Computational Completeness – Proof Ideas

sym(2):

- $p_i : (\mathtt{ADD}(r), p_j)$
  $[_1 p_i \rightarrow p_i [_1$ and $[_1 p_i' p_i'' \rightarrow p_i' p_i'' [_1$
  $p_i' p_j [_1 \rightarrow [_1 p_i' p_j$ and $p_i'' a_r [_1 \rightarrow [_1 p_i'' a_r$
- $p_i : (\mathtt{SUB}(r), p_j, p_k) \in I$
  $[_1 p_i'' p_i \rightarrow p_i'' p_i [_1$ and $[_1 p_i' a_r \rightarrow p_i' a_r [_1$
  $p_i' p_j [_1 \rightarrow [_1 p_i' p_j$ and $p_i'' [_1 \rightarrow [_1 p_i''$ or
  $p_i'' [_1 \rightarrow [_1 p_i''$ and $[_1 p_i' p_i''' \rightarrow p_i' p_i''' [_1$
  $p_i' p_k [_1 \rightarrow [_1 p_i' p_k$ and $p_i''' [_1 \rightarrow [_1 p_i'''$

sym(1), *anti*(1):

- $p_i : (\mathtt{ADD}(r), p_j)$ $a_r [_1 p_i \rightarrow p_i [_1 a_r$ and $p_j [_1 \rightarrow [_1 p_j$
- $p_i : (\mathtt{SUB}(r), p_j, p_k)$
  $p_i'' [_1 p_i \rightarrow p_i [_1 p_i''$ and (eventually!) $p_i' [_1 a_r \rightarrow a_r [_1 p_i'$
  $p_j [_1 p_i'' \rightarrow p_i'' [_1 p_j$ and $[_1 p_i' \rightarrow p_i' [_1$
  $p_i''' [_1 p_i'' \rightarrow p_i'' [_1 p_i'''$ and $p_i' [_1 \rightarrow [_1 p_i'$
  $p_k [_1 p_i''' \rightarrow p_i''' [_1 p_k$ and $[_1 p_i' \rightarrow p_i' [_1$

Computational Completeness – Specific Proof Details

sym(2):
The "garbage" – the symbols p_i', p_i'', p_i''' – cannot be removed from the skin membrane, which does not matter for the accepting case of a P automaton; taking the same construction for generating P systems, this is a challenging question - either we do not care about these "garbage" symbols or we have to add an output membrane.

sym(1), *anti*(1):
Only the final label p_h remains in the skin membrane when the P automaton accepts by halting, hence, by adding the rule $[_1 p_h \rightarrow p_h [_1$ we even end up with an empty skin membrane.
In other words, we could also consider these P automata with minimal antiport and uniport rules as generating mechanisms yielding their results as the numbers of objects in the skin membrane, without having any additional garbage.

Computational Completeness with *min*, min_1

Any partition of rules used for the control of the rules to be applied together consists of only one rule; hence, we can use the same partitioning of rules for the definition of minimal parallelism as well as of 1-restricted minimal parallelism.

Corollary

For $X \in \{N, Ps\}, X \in \{min, min_1\}$,

$$\begin{aligned} XRE &= XO_* P_1 K_* (Y) [sym_2] \\ & \quad XO_* P_1 K_* (Y) [anti_1, sym_1]. \end{aligned}$$

Conclusion

Communication P automata with rule control can accept any recursively enumerable set of (vectors of) natural numbers

- working in the transition modes max, min, min_1 with
- $sym(2)$-rules or $sym(1)$,$anti(1)$ rules
- in only one membrane.

Future Research

- consider the transition modes min_k for $k \geq 2$;
- consider other kinds of rules;
- investigate corresponding generating cases, especially check whether "garbage" symbols can be avoided when using $sym(2)$ rules;
- . . . .

THANK YOU FOR YOUR ATTENTION!

- The P Systems Web Page: `http://ppage.psystems.eu`.