On the Power of Deterministic EC P Systems

Artiom ALHAZOV

Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
E-mail: `artiome.alhazov@estudiants.urv.es`

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
E-mail: `artiom@math.md`

Abstract. It is commonly believed that a significant part of the computational power of membrane systems comes from their inherent non-determinism. Recently, R. Freund and Gh. Păun have considered deterministic P systems, and formulated the general question whether the computing (generative) capacity of non-deterministic P systems is strictly larger than the (recognizing) capacity of their deterministic counterpart.

In this paper, we study the computational power of deterministic P systems in the evolution–communication framework. It is known that, in the generative case, two membranes are enough for universality. For the deterministic systems, we obtain the universality with three membranes, leaving the original problem open.

1 Introduction

We assume the reader familiar with membrane computing (see `http://psystems.disco.unimib.it` for the bibliography). The *evolution–communication* P systems, introduced by M. Cavaliere in [2], are the P systems with two types of rules: simple (i.e., without targets) rewriting rules, and communication (i.e., symport/antiport rules).

A *generative* P system starts from a fixed configuration, and (possibly) halts with a resulting number of objects (or multiset, or a sequence) in a specified region. A *recognizing* P system starts from a fixed configuration plus the input number (or multiset), and the input is accepted if and only if the computation halts.

The purpose of this paper is to prove universality of deterministic recognizing evolution–communication (in short, EC) P systems. In the non-deterministic generative case, EC P systems are known to be universal even when using only two membranes (and symport/antiport rules of a rather small weight). At the price of using one further membrane, we show that the universality holds true also in the deterministic recognizing case; the symport/antiport rules used in the proof are still of a small weight. We do not know whether our results can be improved in the number of membranes.

2 Definitions

A P system is deterministic if for every reachable non-halting configuration the next configuration is unique.

In what follows, we consider P systems which accept numbers: to accept a number N, the system starts with the initial configuration, to which N copies of a specified object a are added in a specified region. The number is accepted if and only if the computation halts. The set of numbers accepted by a system Π is denoted by $N(\Pi)$. The system being deterministic, there is only one computation (either halting, or non-halting) possible for every input.

Let the P system have m membranes and the set O of objects. In this paper, the evolution–communication systems are considered, so the rules (applied in the maximally-parallel manner) are of the following forms:

1. $a \to x$,
   associated to region i, $1 \leq i \leq m$, where $a \in O$, $w \in O^*$,

2. (x, out), (y, in), $(x, out; y, in)$,
   associated to membrane i, where $1 \leq i \leq m$, $x, y \in O^+$.

Thus, the recognizing P system can be denoted as

$$\Pi = (O, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, R'_1, \cdots, R'_m, i_0),$$

where μ is the membrane structure, w_i is the starting multiset of objects in region i, R_i is the set of rules of the first form (evolution), R'_i is the set of rules of the second form (communication), and i_0 is the input region.

By $NOP_m(ncoo, sym_p, anti_q)$ we denote the family of sets $N(\Pi)$ generated by EC P systems with at most m membranes, using non-cooperative evolution rules, symport rules of weight at most p, and antiport rules of weight at most q. When dealing with recognizing (accepting) systems, we add the subscript a to the front N, while, moreover, a D is added in the case of using only deterministic systems. As usual, NRE is the family of Turing computable sets of numbers.

3 The Power

It is known from [1] and [4] that $NOP_2(ncoo, sym_1, ant_1) = NOP_2(ncoo, sym_2) = NRE$. We now present the deterministic counterparts of these results, using 3 membranes.

Theorem 3.1 *$DN_aOP_3(ncoo, sym_1, ant_1) = NRE$.*

Proof. Given a set $M \in NRE$, consider a deterministic register machine $G = (m, e_{init}, e_{halt}, P)$ with m registers, initial label e_{init}, halting label e_{halt}, instruction set P, and set $Lab(P)$ of labels, accepting M. We construct the following P system (object a_i represents the the value of the ith register of G)

$$\begin{aligned}
\Pi &= (O, \mu = [_1[_2[_3]_3]_2]_1, w_1 = \lambda, w_2 = e_{init}, w_3 = \lambda, R_1, R_2, R_3, R'_1 = \emptyset, R'_2, R'_3, 2),\\
O &= \{e, e_0, e_1, e_2, e_3, e_4, e_5, e_6 \mid e \in Lab(P)\}\\
&\cup \ \{a_r \mid 1 \leq r \leq m\} \cup \{s_1, s_2, s_3, q, z\},
\end{aligned}$$

$$
\begin{aligned}
R_1 &= \{s_1 \to s_2\} \cup \{a_r \to \lambda \mid 1 \le r \le m\} \\
&\cup \ \{e_3 \to e_4 \mid (e : dec(r), f, g) \in P\}, \\
R_2 &= \{s_3 \to \lambda\} \cup \{e \to a_r f \mid (e : inc(r), f) \in P\} \\
&\cup \ \{e \to s_1 e_0,\ e_0 \to e_1,\ e_1 \to e_2,\ e_2 \to e_3,\ e_4 \to e_5 q,\ e_5 \to e_6,\ e_6 \to z \\
&\quad \mid (e : dec(r), f, g) \in P\}, \\
R_3 &= \{s_2 \to s_3, q \to \lambda\} \cup \{e_3 \to f \mid (e : dec(r), f, g) \in P\}, \\
R_2' &= \{(s_1, out),\ (a_r, out; s_2, in)\} \\
&\cup \ \{(e_3, out; s_2, in),\ (e_4, in) \mid (e : dec(r), f, g) \in P\}, \\
R_3' &= \{(s_2, in),\ (s_3, out; q, in)\} \cup \{(f, out) \mid (e : dec(r), f, g) \in P\} \\
&\cup \ \{(s_3, out; e_3, in) \mid (e : dec(r), f, g) \in P\}.
\end{aligned}
$$

The P system above recognizes a number N if and only if the computation, starting with a_1^N (the input register of G is the first one) placed in region 2, halts. Below are the simulations of individual instructions.

Instruction $(e : inc(r), f)$ is simulated in the following way:

$$[_1[_2 ew[_3]_3]_2]_1 \Rightarrow [_1[_2 a_r f w[_3]_3]_2]_1.$$

The object e (corresponding to the instruction label) simply evolves into $a_r f$, thus changing instruction label from e to f and adding one to the counter r.

Instruction $(e : dec(r), f, g)$ (in case register r is non-zero) is simulated as follows:

$$
\begin{aligned}
&[_1[_2 e a_r w[_3]_3]_2]_1 \Rightarrow [_1[_2 s_1 e_0 a_r w[_3]_3]_2]_1 \Rightarrow [_1 s_1[_2 e_1 a_r w[_3]_3]_2]_1 \\
&\Rightarrow [_1 s_2[_2 e_2 a_r w[_3]_3]_2]_1 \Rightarrow [_1 a_r[_2 s_2 e_3 w[_3]_3]_2]_1 \Rightarrow [_1[_2 e_3 w[_3 s_2]_3]_2]_1 \\
&\Rightarrow [_1[_2 e_3 w[_3 s_3]_3]_2]_1 \Rightarrow [_1[_2 s_3 w[_3 e_3]_3]_2]_1 \Rightarrow [_1[_2 w[_3 f]_3]_2]_1 \Rightarrow [_1[_2 f w[_3]_3]_2]_1.
\end{aligned}
$$

The object e (corresponding to the instruction label) evolves into e_0 (changing in 3 steps into e_3) and s_1, which goes in region 1, then changes into s_2, and then returns in region 2 in exchange for a_r (which is then erased). Then, s_2 travels into region 3, changes to s_3 and returns to region 2 (where it is then erased) in exchange for e_3. Finally, e_3, being in region 3, changes into f and return in region 2, finishing the simulation of the instruction.

Instruction $(e : dec(r), f, g)$ (in case register r is zero) is simulated as follows:

$$
\begin{aligned}
&[_1[_2 ew[_3]_3]_2]_1 \Rightarrow [_1[_2 s_1 e_0 r w[_3]_3]_2]_1 \Rightarrow [_1 s_1[_2 e_1 w[_3]_3]_2]_1 \\
\Rightarrow\ &[_1 s_2[_2 e_2 w[_3]_3]_2]_1 \Rightarrow [_1 s_2[_2 e_3 w[_3]_3]_2]_1 \Rightarrow [_1 e_3[_2 s_2 w[_3]_3]_2]_1 \\
&\Rightarrow [_1 e_4[_2 w[_3 s_2]_3]_2]_1 \Rightarrow [_1[_2 e_4 w[_3 s_2]_3]_2]_1 \Rightarrow [_1[_2 e_5 q w[_3 s_2]_3]_2]_1 \\
\Rightarrow\ &[_1[_2 e_6 w s_2[_3 q]_3]_2]_1 \Rightarrow [_1[_2 z w[_3]_3]_2]_1.
\end{aligned}
$$

(Note that $|w|_{a_r} = 0$.) Like in the previous case, the object e evolves into e_0 (changing in 3 steps into e_3) and s_1, which goes in region 1, and then changes into s_2. Now there is no object a_r in region 2 to bring s_2 to region 2, so s_2 remains in region 3 until the next step, when it is exchanged with e_3. Then s_2 travels to region 3 and changes into s_3. Now, e_3, being in region 1, changes into e_4, returns to region 2, where it evolves into e_5 (changing it two steps into z) and q, which exchanges with s_3 and then both q and s_3 are erased. □

In the next theorem, symport of weight two is used instead of antiport of weight one, leading to one more universality result.

Theorem 3.2 $DN_aOP_3(ncoo, sym_2) = NRE$.

Proof. Given a set $M \in NRE$, consider a deterministic register machine $G = (m, e_{init}, e_{halt}, P)$ as above, accepting M. We construct the following P system:

$$\begin{aligned}
\Pi &= (O, \mu = [_1[_2[_3]_3]_2]_1, w_1 = \lambda, w_2 = e_{init}, w_3 = \lambda, R_1, R_2, R_3, R'_1 = \emptyset, R'_2, R'_3, 2), \\
O &= \{e, e_0, e_1, e_2, e_3, e_4, e_5, e_6 \mid e \in Lab(P)\} \\
&\cup \{a_r \mid 1 \le r \le m\} \cup \{s_1, s_2, s_3, q, z\}, \\
R_1 &= \{q \to \lambda, s_2 \to \lambda\} \cup \{e_0 \to e_1 \mid (e : dec(r), f, g) \in P\} \\
&\cup \{a_r \to \lambda \mid 1 \le r \le m\}, \\
R_2 &= \{s_1 \to s_2\} \cup \{e \to s_1e_0,\ e_1 \to e_2q,\ e_2 \to e_3,\ e_3 \to f \mid (e : dec(r), f, g) \in P\} \\
&\cup \{e \to a_rf \mid (e : inc(r), f) \in P\}, \\
R_3 &= \{s_2 \to \lambda\} \cup \{e_0 \to z \mid (e : dec(r), f, g) \in P\}, \\
R'_2 &= \{(qs_2, out)\} \cup \{(e_0a_r, out),\ (e_1, in) \mid (e : inc(r), f) \in P\}, \\
R'_3 &= \{(s_2e_0, in),\ (z, out) \mid (e : dec(r), f, g) \in P)\}.
\end{aligned}$$

The P system above recognizes a number N if and only if the computation, starting with a_1^N (the first register is the input one of G) placed in region 2, halts. Below is the simulation of the instructions of G.

Instruction $(e : inc(r), f)$ is simulated like in the previous theorem:

$$[_1[_2ew[_3]_3]_2]_1 \Rightarrow [_1[_2Rfw[_3]_3]_2]_1.$$

Instruction $(e : dec(r), f, g)$ is simulated in the following way: The object e evolves in e_0 (used to subtract) and s_1 (which changes into s_2, the helper).

$$\begin{aligned}
&[_1[_2ea_rw[_3]_3]_2]_1 \Rightarrow [_1[_2s_1e_0a_rw[_3]_3]_2]_1 \Rightarrow [_1e_0a_r[_2s_2w[_3]_3]_2]_1 \\
\Rightarrow\ &[_1e_1[_2s_2w[_3]_3]_2]_1 \Rightarrow [_1[_2e_1s_2w[_3]_3]_2]_1 \Rightarrow [_1[_2e_2qs_2w[_3]_3]_2]_1 \\
\Rightarrow\ &[_1qs_2[_2e_3w[_3]_3]_2]_1 \Rightarrow [_1[_2fw[_3]_3]_2]_1.
\end{aligned}$$

If a_r is present in region 2, then (one copy of) a_r goes to region 1 (where it is erased) together with e_0, which changes into e_1, returns to region 2, and then evolves into e_2 (which changes into f in two steps) and q, which exists to region 1 together with s_2, where both q and s_2 are erased.

$$\begin{aligned}
&[_1[_2ew[_3]_3]_2]_1 \Rightarrow [_1[_2s_1e_0w[_3]_3]_2]_1 \Rightarrow [_1[_2s_2e_0w[_3]_3]_2]_1 \\
\Rightarrow\ &[_1[_2w[_3s_2e_0]_3]_2]_1 \Rightarrow [_1[_2w[_3z]_3]_2]_1 \Rightarrow [_1[_2zw[_3]_3]_2]_1.
\end{aligned}$$

(Note that $|w|_{a_r} = 0$.) If a_r is not present in region 2, then e_0 waits for s_2, they both come to region 3, where s_2 is erased, while e_0 changes to z and returns to region 2, finishing the simulation of the instruction. □

4 Conclusions

This paper gives two three-membrane constructions for the universal deterministic evolution–communication P systems, one using symport of weight at most two, and the other one using symport and antiport of weight one. These results are incomparable with the existing (nondeterministic) universality results with two membranes, as the proofs rely on having three regions where evolution rules take place. It is *an open question* whether the EC P systems with two membranes are universal in the deterministic way with symport of weight at most two, or with symport and antiport of weight one.

Acknowledgements. The author acknowledges IST-2001-32008 project "MolCoNet", as well as the Moldovan Research and Development Association (MRDA) and the U.S. Civilian Research and Development Foundation (CRDF), Award No. MM2-3034 for providing a challenging and fruitful framework for cooperation.

References

[1] A. Alhazov, Minimizing Evolution-Communication P Systems and EC P Automata, *Brainstorming Week on Membrane Computing* (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), Rovira i Virgili University, Technical Report **26/03**, Tarragona, 2003, 23–31, and *New Generation Computing*, accepted for publication.

[2] M. Cavaliere, Evolution-Communication P Systems, *Membrane Computing. International Workshop, WMC-CdeA 2002*, Curtea de Argeş (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), Springer-Verlag, LNCS **2597**, Berlin, 2003, 134–145.

[3] R. Freund, Gh. Păun, On Deterministic P Systems, submitted, 2003.

[4] S.N. Krishna, A. Păun, Some Universality Results on Evolution-Communication P Systems, *Brainstorming Week on Membrane Computing* (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), Rovira i Virgili University, Technical Report **26/03**, Tarragona, 2003, 207–215.

[5] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.