The Power of Symport-3 with Few Extra Symbols

Artiom Alhazov[1,2] and Yurii Rogozhin[2]

[1] Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Viale Sarca 336, 20126 Milano, Italy
E-mail: artiom.alhazov@unimib.it

[2] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
E-mail: {artiom,rogozhin}@math.md

Summary. Membrane systems (with symbol objects) are formal models of distributed parallel multiset processing. Symport rules move multiple objects to a neighboring region. It is known that P systems with symport rules of weight at most 3 and a single membrane are computationally complete with 7 superfluous symbols. It is also known that without any superfluous symbols such systems only generate finite sets.

We improve the lower bounds on the generative power of P systems with few superfluous objects as follows. 0: empty set and all singletons; k: all sets with at most k elements and all sets of numbers k+regular with up to k states, $1 \leq k \leq 5$; 6: all regular sets of non-negative integers. All results except the last one are also valid for different modes, e.g., sequential one, also for higher values of k.

1 Introduction

Membrane systems (with symbol objects) are formal models of distributed parallel multiset processing. Symport rules move predefined groups objects to a neighboring region [4]. In maximally parallel mode (typical for membrane computing), this alone is sufficient to construct a computationally universal device, as long as the environment may contain an unbounded supply of some objects. The number of symbols specified in a symport rule is called its weight. The result of a computation is the total number of objects when the system halts. In some cases, however, for technical reasons the desired result may only be obtained alongside a small number of superfluous objects in the output region.

There were multiple papers improving the results on P systems with symport/antiport of small weight (an antiport rule moves objects between 2 regions in both directions, and its weight is the maximum of objects per direction), see [2] for a survey of results. Computational completeness is achieved even for minimal cooperation: either symport/antiport of weight 1, or symport of weight at most

2. This holds for 2 membranes, without superfluous objects if the output is considered in the skin, or with 1 superfluous object under the classical assumption of the output in the elementary membrane. In the tissue case, the accepting systems can even be made deterministic.

With cooperation of up to 3 objects, a single membrane suffices. The regions are called the skin and the environment, the latter contains an unbounded supply of some objects, while the contents of the former is always finite. With antiport-2/1 alone (i.e., exchanging 1 object against 2), the computational completeness is obtained with a single superfluous object. With symport-3 (i.e., symport rules only, of weight up to 3), one proved in [3] that 13 extra objects suffices for computational completeness. This result has been improved in [1] to 7 superfluous symbols. In the same paper it was shown that without any superfluous symbols such systems only generate finite sets.

The computation consists of multiple, sometimes simultaneous, actions of two types: move objects from the skin to the environment, and move objects from the environment into the skin. It is obvious that trying to move all objects out in the environment will activate the rules of the second type. Since, clearly, rules of the first type alone cannot generate more than finite sets, it immediately follows that the "garbage" is unavoidable. This paper tries to improve the currently best bounds on how much "garbage" is sufficient.

2 Definitions

Throughout the paper, by "number" we will mean a non-negative integer. We write N_jFIN_k to denote the family of all sets of numbers each not smaller than j, of cardinality k. By N_jREG_k we denote the family of all sets M of numbers each not smaller than j, such that $\{x - j \mid x \in M\}$ is accepted by some finite automaton with k states, with at least one transition from every non-final state. We assume the reader to be familiar with the basics of the formal language theory, and we recall that for a finite set V, the set of words over V is denoted by V^*, the set of non-empty words is denoted by V^+, and a multiset may be represented by a string, representing the multiplicity by the number of occurrences, the order not being important.

2.1 Finite Automata

Definition 1. *A finite automaton is a tuple $A = (\Sigma, Q, q_0, \delta, F)$, where Σ is an input alphabet, Q is the set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition mapping.*

The function δ is naturally extended from symbols to strings. The language accepted by A is the set $\{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$.

2.2 P Systems with Symport

The scope of this paper is limited to P systems with symport only, with a single membrane.

Definition 2. *A P system with symport rules and one membrane is a tuple*

$$\Pi = (O, E, \mu = [\ ]_1, w, R), \text{ where}$$

- O *is a finite set called alphabet; its elements are called symbols,*
- $E \subseteq O$ *is the set of objects appearing in the environment in an unbounded supply,*
- μ *is the membrane structure, trivial in case of one membrane; the inner region is called the skin and the outer region is called the environment,*
- $w \in O^*$ *is the specification of the initial contents of the inner region,*
- R *is the set of rules of types* (u, out) *or* (v, in), $u, v \in O^+$.

An action of a rule (u, out) is to move the multiset of objects specified by u from the skin into the environment. An action of a rule (v, in) is to move the multiset of objects specified by v from the environment into the skin ($v \in E^*$ is not allowed by definition). The objects are assigned to rules non-deterministically. Maximal parallelism allows application of multiple rules simultaneously and multiple times, as long as there are enough copies of objects for them, and provided no further rule is applicable to the unassigned objects.

The computation halts when no rules are applicable at some step. The result of a halting computation is the total number of objects in the inner region when it halts. The set of numbers $N(\Pi)$ generated by a P system Π is the set of results of all its computations.

The family of sets of numbers generated by a family of P systems with one membrane and symport rules of weight at most k is denoted by $NOP_1(sym_k)$ in maximally parallel mode. We add superscript *sequ* to P to indicate sequential mode instead.

3 Few-Element Sets

We now present a few simple systems.

$$\Pi_0 = (O = \{a\}, E = \emptyset, \mu = [\ ]_1, w = a, R = \{(a, in), (a, out)\}).$$

System Π_0 perpetually moves a single object in and out, effectively generating the emptyset. For any $x \in \mathbb{N}$, setting $R = \emptyset$ and $w = a^x$ will lead to a system Π_1 which immediately halts, generating a singleton $\{x\}$.

We now proceed to arbitrary small-cardinality sets. To generate a multi-element set, the system must make at least one non-deterministic choice. Since we want to allow the difference between the elements to be arbitrarily large, such choice

must be persistent, i.e., the decision information should not vanish, at least until multiple objects are moved accordingly. For any numbers $y > x$, consider the following P system:

$$\Pi_2 = (O = \{a, b, i, p, q\}, E = \{q\}, \mu = [\ ]_1, w = a^x b^{y-x+1} ip, R),$$
$$R = \{(i, out), (ip, out), (pq, in), (pqb, out)\}.$$

There are two possible computations of Π_2: either i exits alone, halting with $a^x b^{y-x+1} p$, generating $y+2$, or i exits with p, leading to a sequence of application of the last two rules until no objects b remain in the skin, halting with $a^x pq$, generating $x+2$. Therefore, Π_2 generates an arbitrary 2-element set with 2 extra objects.

This construction can be improved to generate higher-cardinality sets as follows. Let $m \geq 2$; for arbitrary $m+1$ distinct numbers denote the largest one by y and the others by x_j, $1 \leq j \leq m$. We construct another P system:

$$\begin{aligned}
\Pi_{m+1} &= (O, E = \{q_j \mid 1 \leq j \leq m\}, \mu = [\ ]_1, w, R),\\
O &= \{a_j \mid 1 \leq j \leq y+1\} \cup \{i\} \cup \{p_j, q_j \mid 1 \leq j \leq m\},\\
w &= i \prod_{j=1}^{y+1} a_j \prod_{j=1}^{m} p_j,\\
R &= \{(i, out)\} \cup \{(ip_j, out), (p_j q_j, in) \mid 1 \leq j \leq m\}\\
&\quad \cup (p_j q_j a_k, out) \mid 1 \leq j \leq m,\ x_j + 1 \leq k \leq y+1, j \neq k\}.
\end{aligned}$$

Such system behaves like Π_2, except it also chooses among different objects p_j to send out symbols a_k for $k > x_j$. It halts either with $a_1 \cdots a_{y+1} p_1 \cdots p_m$ generating $y+m+1$, or with $a_1 \cdots a_{x_j} q_j p_1 \cdots p_m$ generating $x_j + m + 1$. For $m = 2, 3, 4$ this leads to Π_3 generating $\{x_1+3, x_2+3, y+3\}$, Π_4 generating $\{x_1+4, x_2+4, x_3+4, y+4\}$, and Π_5 generating $\{x_1+5, x_2+5, x_3+5, x_4+5, y+5\}$, i.e., any 3-, 4- or 5-element set with 3, 4 or 5 extra objects, respectively.

4 Sequential Mode and Straightforward Regularity

Remark 1. All P systems constructed in the previous section de facto work sequentially, hence the results are valid for P system in any other traditional derivation mode (e.g., sequential, asynchronous, minimally parallel, maximal strategy).

The sequential mode is an interesting candidate for a research topic, due to its simplicity, even though its power (as that of any sequential multiset rewriting system without control) cannot exceed $NMAT = NREG$, and infinite sets without zero cannot be generated in this case either, for the same argument as in maximally parallel case.

We now proceed by constructing a P system generating the length set of a language accepted by a finite automaton $A = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_j \mid$

$0 \leq j \leq m\}$; we assume A satisfies the following property: there is at least one transition from every non-final state.

$$\begin{aligned}\Pi_A = (O = Q \cup Q' \cup \Sigma, E = Q' \cup \Sigma, \mu = [\ ]_1, w = q_0 \cdots q_m q'_0, R),\\ R = \{(q_j q'_j, out) \mid 0 \leq j \leq m\}\\ \cup \{(q_j a q'_k, in) \mid q_k \in \delta(q_j, a),\ a \in \Sigma\} \cup \{(q'_j, out) \mid q_j \in F\}.\end{aligned}$$

Notice also that adding an arbitrary number of objects from Σ to the initial configuration increases the result by the corresponding number. Unfortunately, besides the needed number, the skin region at halting also contains the superfluous symbols, as many as there are states in A. Therefore, we have obtained all sets $N_j REG_k$, $j \geq k$. Within this section it is enough to consider $j = k$, because REG_j contains REG_k.

Remark 2. If we fix j, k and restrict A to be deterministic, then the number set M generated by the corresponding P system can be characterized by the following properties: $x \geq j$ for all $x \in M$ and there exists a number $0 \leq p \leq k$ such that if $x \geq j + k - p$, then $x \in M$ if and only if $x + p \in M$. Hence, acceptability of sufficiently large numbers is determined by the remainder of their division by some $p < k$. The general result is also valid for non-deterministic systems, but exact characterization in terms of states is less straightforward.

The simplest examples of application of Π_A are the set of all positive numbers and the set of all positive even numbers.

It is not difficult to notice that the result of Π_A does not depend on the computation mode, e.g., it is valid also for maximally parallel mode. Therefore, both $NOP_1^{sequ}(sym_3)$ and $NOP_1(sym_3)$ contain

$$NFIN_0 \cup NFIN_1 \cup \bigcup_{k=2}^{\infty} N_k FIN_k \cup \bigcup_{k=1}^{\infty} N_k REG_k.$$

We recall that the upper bound for $NOP_1^{sequ}(sym_3)$ is $N_1 REG \cup NFIN$.

5 Larger Sets and Few Extra Objects

We now focus on the maximally parallel mode, and revisit the symport-3 construction from [1]. The 7 extra objects were denoted $l_h, b, d, x_1, x_4, x_5, x_6$.

Recall that any regular set of numbers is generated by some non-deterministic register machine with only ADD-instructions, or, equivalently, accepted by a finite automaton. It suffices to take the same construction, and notice that object d is no longer needed to check for the conflicting counters. Removing it from the system leads to P systems generating $N_6 REG$. Hence, the contribution of the previous

section for maximally parallel P systems can be limited to infinite sets of the form small number+infinite regularity accepted by at most 5 states.

We now recall the construction from [1] with the corresponding changes to generate N_6REG, rewritten in the syntax of finite automata. Let L be an arbitrary set from N_6REG. Then there exists a finite automaton $A = (\Sigma, Q, q_0, \delta, F)$ accepting $L - 6 = \{n - 6 \mid n \in L\}$. We construct a P system simulating A:

$$
\begin{aligned}
&\Pi = (O, E, [\ ]_1, w, R, 1), \text{ where}\\
&O = \{x_i \mid 1 \le i \le 6\} \cup Q \cup \{(p,q,j) \mid p,q \in Q,\ 1 \le j \le 3\} \cup \{a, A, b, H\},\\
&E = Q \cup \{(p,q,2) \mid p,q \in Q\} \cup \{a, A, x_2, x_3, H\},\\
&w = q_0 x_1 x_4 x_5 x_6 b A \prod_{p,q \in P} (p,q,1)(p,q,3),\\
&R = \{1 : (x_1 x_2 x_3, in),\ 2 : (x_2 x_4 x_5, out),\ 3 : (x_3 x_6, out),\ 4 : (Hb, in)\}\\
&\quad \cup \{5 : (qb, out) \mid q \in F\}\\
&\quad \cup \{6 : (Hbx, out) \mid x \in \{(p,q,1),(p,q,3) \mid p,q \in Q\} \cup \{A\}\}\\
&\quad \cup \{7 : (p(p,q,1)x_1, out),\ 8 : ((p,q,1)x_4(p,q,2), in),\\
&\qquad 9 : ((p,q,2)(p,q,3)A, out), 10 : ((p,q,3)x_5 q, in),\\
&\qquad 11 : (Ax_6 a, in) \mid q \in \delta(p,s)\}.
\end{aligned}
$$

Notice that the effect of rules 1,2,3 is that sending object x_1 out will bring x_2 and x_3 in, which, in turn, will send objects x_4, x_5 and x_6 out. Notice also that rules 8,10,11 need x_4, x_5 and x_6, so if these objects are inside the membrane, then all objects of the form (p,q,j), $j \in \{1,3\}$ may be sent out, without enabling these rules. We will skip mentioning objects x_j in the simulation.

The simulation of a transition in A is performed as follows:

- The state p brings object $(p,q,1)$ out, also sending x_1 out to enable the rest of the simulation.
- Object $(p,q,1)$ brings object $(p,q,2)$ in, which, in turn, brings both $(p,q,3)$ and A out.
- Object $(p,q,3)$ brings in the next state q, while object A brings in object a, contributing to the result.

If the current state is final, rule 5 may be applied, leading to iteration of rules 4 and 6, taking out all objects except H, b, x_1, x_4, x_5, x_6 and the desired number of copies of a.

Remark 3. If we were interested in generating vectors rather than numbers, simulation of partially blind register machines could be also performed with six additional objects. Since this gives no additional power for numbers (i.e., NMAT=NREG), we presented the simpler construction, without subtraction.

6 Discussion

It has been known that P systems with symport rules of weight at most 3 generate at least N_7RE, and cannot generate infinite sets containing 0. We have improved the lower bound to

$$NFIN_0 \cup NFIN_1 \cup N_2FIN_2 \cup N_3FIN_3 \cup N_4FIN_4 \cup N_5FIN_5 \\ \cup N_1REG_1 \cup N_2REG_2 \cup N_3REG_3 \cup N_4REG_4 \cup N_5FIN_5 \\ \cup N_6REG \cup N_7RE.$$

It is open whether this bound is tight, since the current best known upper bound is $N_1RE \cup NFIN$.

For the sequential case, the bounds are given in the end of Section 4. It is particularly interesting whether infinitely many additional objects are unavoidable for generation of regular number sets in the sequential mode.

Acknowledgements

The first author acknowledges the project RetroNet by the Lombardy Region of Italy under the ASTIL Program (regional decree 6119, 20100618).

References

1. A. Alhazov, R. Freund, Yu. Rogozhin: Computational Power of Symport/Antiport: History, Advances and Open Problems. In: R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa: *Membrane Computing, 6th International Workshop*, WMC 2005, Vienna, Revised Selected and Invited Papers. *Lecture Notes in Computer Science* **3850**, Springer, 2006, 1–30.
2. R. Freund, A. Alhazov, Yu. Rogozhin, S. Verlan: Communication P Systems. Chapter 5 in: Gh. Păun, G. Rozenberg, A. Salomaa: *The Oxford Handbook of Membrane Computing*, 2010, 118–143.
3. P. Frisco, H.J. Hoogeboom: P Systems with Symport/Antiport Simulating Counter Automata. *Acta Informatica* **41**, 2004, 145-170.
4. A. Păun, Gh. Păun: The Power of Communication: P Systems with Symport/Antiport. *New Generation Computing* **20**, 2002, 295-305.