ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **13**, Number 2, 2010, 113–130

On Communication Complexity in Evolution-Communication P Systems

Henry ADORNA[1], Gheorghe PĂUN[2,3], Mario J. PÉREZ-JIMÉNEZ[3]

[1]Department of Computer Science (Algorithms and Complexity)
University of the Philippines-Diliman
1101 Quezon City, Philippines
E-mail: hnadorna@dcs.upd.edu.ph

[2]Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania

[3] Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: gpaun@us.es, marper@us.es

Abstract. Looking for a theory of communication complexity for P systems, we consider here so-called evolution-communication (EC for short) P systems, where objects evolve by multiset rewriting rules without target commands and pass through membranes by means of symport/antiport rules. We first propose a way to measure the communication costs by means of "quanta of energy" (produced by evolution rules and) consumed by communication rules. EC P systems with such costs are proved to be Turing complete in all three cases with respect to the relation between evolution and communication operations: priority of communication, mixing the rules without priority for any type, priority of evolution (with the cost of communication increasing in this ordering in the universality proofs).

More appropriate measures of communication complexity are then defined, as dynamical parameters, counting the communication steps or the number (and the weight) of communication rules used during a computation. Such parameters can be used in three ways: as *properties* of P systems (considering the families of sets of numbers generated by systems with a given communication complexity), as *conditions* to be imposed on computations (accepting only those computations with a communication complexity bounded by a given threshold), and as standard *complexity measures* (defining the class of problems which can be solved by P systems with a bounded complexity). Because we ignore the evolution steps, in all three cases it makes sense to consider hierarchies starting with finite complexity thresholds. We only give some preliminary results about these hierarchies (for instance, proving that already their lower levels contain complex – e.g., non-semilinear – sets), and we leave open many problems and research issues.

1. Introduction

Although membrane computing can still be considered a young branch of natural computing, [12], the theory of P systems is well developed (see details in [16] and at the area website from [20]). In particular, many research efforts were devoted to complexity issues related to P systems. First, all results dealing with the number of membranes, with normal forms, with various static parameters (estimating the size of P systems from various points of view) can be considered as contributing to the *descriptional complexity* of P systems (this was a much investigated topic in formal languages theory, see, e.g., [6]). Much more coherently developed, forming an explicit theory, is the *computational complexity* of P systems, where the main complexity parameter is *time*, the number of steps of a computation. In this framework, several specific complexity classes were defined, elaborate relations with the $\mathbf{P} \neq \mathbf{NP}$ conjecture were found, as well as many other related results; we refer to the recent survey [18] and to the corresponding chapter of [16] for details. Recently, also a *space* complexity theory for P systems was initiated – see, e.g., [19].

What is still almost not at all considered is the *communication complexity* of P systems, in spite of the fact that this was suggested as a research topic several times (for instance, in [3] and [14], but without any hint about a possible definition). A related measure is discussed in [4] for symport/antiport P systems, the so-called "communication difference", denoted $Comdif$, defined as the difference of numbers of input and output objects in an antiport rule, but this is again a static measure, defined for rules of a P system.

We address this issue here, with several basic proposals, but our preliminary results indicate that this is a non-trivial research direction – especially if we want to get close to the classic theory of communication complexity, as synthesized, e.g., in [8]. Roughly speaking, the classic framework deals with a distributed/parallel computing device and complex problems which are split into subproblems and the parts are distributed to separate "processors", which cooperate in solving the general problem; to this aim, the processors need to communicate and the number of bits used to this aim gives the communication complexity of the solution. P systems are distributed and parallel devices, but we do not have an explicit protocol for solving problems in a distributed manner, after introducing subproblems of a problem in various subsystems; we have various tools for communication among membranes, but the amount of communication was not yet explicitly and systematically investigated.

The present paper mainly calls once again the attention to this research direction, as we do not propose yet a way to solve problems in a distributed way using a P system, so that a framework as that in [8] to be obtained. Instead, we first go back to a sort of a descriptional complexity measure, defined for evolution-communication (EC for short) P systems as introduced in [1] which was then investigated in a series of papers, e.g., in [9]. In these systems, the evolution of objects is separated from their communication across membranes: the evolution is done by means of multiset rewriting rules and the communication by symport/antiport rules. In order to evaluate the communication effort, we consider a *cost* of using a communication rule, in the form of a quantity of "energy" consumed by that rule. Specifically, we consider a special object, e, called "quantum of energy"; evolution rules can produce amounts of energy, which is consumed by the communication rules. Three types of EC P systems are investigated: with priority of communication on evolution, with steps where rules of the two types are non-deterministically mixed, and with priority of evolution on communication. In all cases, universality results are obtained (this was known only for the intermediate case, of no priority among the two types of operations), but, interesting enough, the cost of communication increases in the order we have mentioned the three cases: one quantum per rule, three quanta per rule, and five quanta per rule, respectively, are used in the corresponding proofs.

We note that P systems with the use of rules controlled by energy were considered already, e.g., in [17], [10], [11], but in different frameworks (other types of P systems, different goals).

A natural step is then to pass from this kind of a static measure to some dynamical ones, with natural definitions: count the communication steps, or the communication rules, or all quanta of energy used during all steps of a halting computation. An infinite cost (hence infinitely many communication rules used during the computation) leads to universality, but also considering only a finite number of communication steps (of communication rules used during a computation) makes sense. Actually, counting only the number of communication steps is nothing else than the length of the computation (the time complexity) . . . ignoring the evolution steps. This means that the computation can be arbitrarily complex in terms of evolution steps (we however use here only non-cooperative evolution rules), what is counted are the communication rules. Such a parameter can be investigated from three points of view: as a *property* of P systems, as a *condition* for selecting only certain computations as acceptable, as the *effort* to solve (decision) problems. In all cases, we can start with finite thresholds (no communication step, one communication step, and so on), hence infinite hierarchies are expected. We only give some hints about the possible proofs of the infinity of these hierarchies, as well as examples showing that already the lower levels of the hierarchies contain complex sets of numbers (e.g., non-semilinear). Thus, besides definitions and preliminary results, we provide here mainly open problems and research topics.

2. Some Prerequisites

Before introducing the class of P systems we investigate in this paper, let us fix some notation and terminology.

The free monoid generated by an alphabet V is denoted with V^* and its neutral element (the empty string) is denoted by λ; the set $V^* - \{\lambda\}$ (of non-empty strings over V) is denoted by V^+. For $a \in V, x \in V^*$, $|x|$ denotes the length of x, and $|x|_a$ is the number of occurrences of symbol a in the string x.

The families of semilinear and of recursively enumerable sets of vectors of dimension $k \geq 1$ of natural numbers are denoted by $SLIN^k$ and N^kRE, respectively; if $k = 1$ (hence we deal with numbers, vectors of one dimension), then the superscript is omitted.

In the proofs of our universality results we use the notion of a (non-deterministic) *register machine*, which is a device $M = (m, B, l_0, l_h, R)$, where $m \geq 1$ is the number of registers, B is the (finite) set of instruction labels, l_0 is the initial label, l_h is the halting label, and R is the finite set of instructions labeled (uniquely identified, with each label being associated with only one instruction) by elements from B. The labeled instructions are of the following forms:

– $l_i : (\texttt{ADD}(r), l_j, l_k)$, $1 \leq r \leq m$ (add 1 to register r and go nondeterministically to one of the instructions with labels l_j, l_k),
– $l_i : (\texttt{SUB}(r), l_j, l_k)$, $1 \leq r \leq m$ (if register r is not empty, then subtract 1 from it and go to the instruction with label l_j, otherwise go to the instruction with label l_k).

A register machine generates a set of natural numbers in the following manner: we start computing with all m registers empty, with the instruction labeled l_0; if the label l_h is reached, then the computation *halts* and the value of register 1 is the number generated by the computation (without loss of generality, all other registers can be assumed to be empty at that time). The set of all natural numbers generated in this way by M is denoted by $N(M)$. It is known that non-deterministic register machines (with three registers) can generate any set of Turing computable sets of natural numbers. If the contents of several registers is taken into consideration in the end of a computation, then vectors of natural numbers are generated.

Convention: When comparing two number generating devices, number 0 is omitted.

3. Evolution-Communication P Systems

Although the notions we work with are introduced below, it would be useful if the reader has some familiarity with basic facts of membrane computing.

The class of P systems which we investigate in this paper is that of EC P systems introduced in [1]. Such a system is a construct of the form

$$\Pi = (O, \mu, w_1, \ldots, w_m, R_1, R'_1, \ldots, R_m, R'_m, i_{out}),$$

where O is the alphabet of objects, μ is the membrane structure (with m membranes, organized in a hierarchical manner, hence with the membrane structure described by a tree, and given as an expression of labeled parentheses; in this definition, like in most cases in the paper, the membranes are labeled with natural numbers, but any alphabet of labels may also be used), $w_1, \ldots, w_m$ are (strings over O representing) 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 evolution rules associated with the regions of μ, $R'_1, \ldots, R'_m$ are finite sets of symport/antiport rules associated with the membranes of μ, and i_{out} is the label of the output membrane. (Note that the evolution rules are associated with the regions and the communication rules are associated with the membranes.) The number m of membranes in μ is called the *degree* of Π. The evolution rules are of the form $a \to v$, where $a \in O, v \in O^*$ (hence non-cooperative and without target indications in the right hand side, as usual in transition P systems), while the symport/antiport rules are of the standard forms used in symport/antiport P systems ($(u, in), (u, out)$ for symport rules and $(u, out; v, in)$ for antiport rules, where $u, v \in O^+$; the length of u in a symport rule and the maximum of $|u|, |v|$ in an antiport rule is called the *weight* of the rule, with the maximum degree over all rules being called the weight of the system).

The weight of symport and antiport rules gives already an indication on the communication complexity of the system. Here we stress this, in the following way. We allow only minimal symport and antiport rules, hence with the multisets u, v above consisting of single objects in O, and, moreover, we add to the system a special object, e, which does not belong to O and can appear in rules as follows. The evolution rules can be of the form $a \to v$, with $a \in O, v \in (O \cup \{e\})^*$, and the symport/antiport rules can be of the forms $(ae^i, in), (ae^i, out)$, where $a \in O, i \geq 1$, and $(ae^i, out; be^j, in)$, where $a, b \in O$ and $i, j \geq 0, i + j \geq 1$. Note that the "quantum of energy" e can be produced by evolution rules and that no communication can be done without involving an amount of "energy". Actually, this energy is consumed by the communication rules: after using a symport or antiport rule, the objects of O are transported across the membrane with which the rule is associated and the occurrences of object e are lost, they do not pass from a region to another one. In the proofs from the next section we will discuss in some detail the way the communication rules are applied, so we do not give here any example.

When specifying a system Π, we explicitly mention e as a component of the tuple, after O.

In a symport rule as above, the number i is called the *energy* of the rule, while in an antiport rule as above the sum $i + j$ is called the *energy*.

In [1] (and [9]), the rules of an EC P systems are used in a non-deterministic maximally parallel way, without any separation of evolution and communication operations. Here we consider three cases (we call them *modes*): (i) communication has priority over evolution (indicated by CPE) – if a communication rule is applicable in any membrane of the system, then at this step only communication rules are used (in the non-deterministic maximally parallel way), and no evolution rule is applied in the system; (ii) communication and evolution rules are applied together, mixed (indicated by CME), as in [1]; (iii) evolution has priority over communication (indicated by EPC) – if any evolution rule can be used in the system, then only such rules are used at that step, and no communication operation is performed.

Using the rules in the non-deterministic maximally parallel way, in one of the three modes suggested above, we obtain transitions among configurations of the system, then computations and halting computations as usual in membrane computing. In the next section we consider the set $N_{mode}(\Pi)$ of numbers generated by a P system Π, by counting the objects of O present in region i_{out} in the halting configuration of computations in Π, performed in the in the mode $mode \in \{CPE, CME, EPC\}$. The family of sets of numbers $N_{mode}(\Pi)$ generated in this way by EC P systems with at most $m \geq 1$ membranes, symport rules of maximal energy $p \geq 0$ and antiport rules of maximal energy $q \geq 0$, is denoted by $N_{mode}ECP_m(sym_p, anti_q)$; when one of the parameters m, p, q is not bounded, we replace the respective subscript with $*$.

4. The Power of EC P Systems with Energy

We start by pointing out some inclusions which directly follow from the definitions:

Lemma 4.1. *$N_{mode}ECP_m(sym_p, anti_q) \subseteq N_{mode}ECP_{m'}(sym_{p'}, anti_{q'}) \subseteq NRE$, for all $1 \leq m \leq m'$, $0 \leq p \leq p'$, $0 \leq q \leq q'$, and $mode \in \{CPE, CME, EPC\}$; each of m', p', q' can also be equal to $*$.*

Actually, most of the inclusions above are equalities.

Theorem 4.1. *$N_{CPE}ECP_m(sym_p, anti_q) = NRE$, for all $m \geq 4, p \geq 1, q \geq 0$.*

Proof. We only prove the inclusion $NRE \subseteq N_{CPE}ECP_4(sym_1, anti_0)$, and to this aim we use the characterization of NRE by means of register machines with three registers. Let $M = (3, B, l_0, l_h, R)$ be such a machine. We construct an EC P system

$$\Pi = (O, e, [\ [\ \ ]_1 [\ \ ]_2 [\ \ ]_3\ ]_0, l_0', \lambda, \lambda, \lambda, R_0, R_0', R_1, R_1', R_2, R_2', R_3, R_3', 1),$$

with $O = \{l, l', l'', l''', \bar{l} \mid l \in B\} \cup \{a, \#\}$, and the sets of rules as mentioned in the following tables, which show the way the system Π simulates the instructions of M (note that initially we have the primed version of the initial label of M present in the skin region and nothing else in the whole system).

For any ADD instruction of the form l_i : $(\texttt{ADD}(r), l_j, l_k)$ in R, we perform the following five steps in Π (we indicate for each step the rules associated with membranes/regions 0 and r):

Step	R_0	R_r	R_r'
1	$l_i' \to l_i e$	–	–
2	–	–	$(l_i e, in)$
3	–	$l_i \to \bar{l}_i a e$	–
4	–	–	$(\bar{l}_i e, out)$
5	$\bar{l}_i \to l_s', s = j, k$	–	–

We also add the rules

$$\bar{l}_i \to \#,\ \# \to \#$$

to R_r (they are used in steps 5 and 6 (and then $\# \to \#$ forever) if in step 4 one uses the rule (ae, out) which is present in R_r' if there are SUB instructions for this register – see below).

The simulation of the ADD rule is obvious – and the fact that the communication has priority over evolution plays no role here as only one (type of) rule can be applied in any step.

The simulation of a SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ in R is more intricate. We first indicate the rules present in sets R_0, R_r, R'_r:

$$\begin{aligned} R_0 &= \{l'_i \to l_i e,\ l'''_i \to l'_j,\ l''_i \to l'_k\}, \\ R_r &= \{l_i \to l'_i e,\ l'_i \to l''_i,\ l''_i \to l'''_i e\}, \\ R'_r &= \{(l_i e, in),\ (ae, out),\ (l'''_i e, out),\ (l''_i e, out)\}. \end{aligned}$$

The way these rules are used for simulating the SUB instruction in the case when the register r is non-empty is shown in the next table:

Step	R_0	R_r	R'_r
1	$l'_i \to l_i e$	–	–
2	–	–	$(l_i e, in)$
3	–	$l_i \to l'_i e$	–
4	–	–	(ae, out)
5	–	$l'_i \to l''_i$	–
6	–	$l''_i \to l'''_i e$	–
7	–	–	$(l'''_i e, out)$
8	$l'''_i \to l'_j$	–	–

The simulation of the SUB instruction in the case of an empty register r is indicated below:

Step	R_0	R_r	R'_r
1	$l'_i \to l_i e$	–	–
2	–	–	$(l_i e, in)$
3	–	$l_i \to l'_i e$	–
4	–	$l'_i \to l''_i$	–
5	–	–	$(l''_i e, out)$
6	$l''_i \to l'_k$	–	–

The first three steps are identical. If register r is non-empty, hence membrane r contains objects a, then, because the communication has priority, one copy of a is removed from membrane r and the rule $l'_i \to l''_i$ is used in the next step. If the register is empty, then the rule $l'_i \to l''_i$ is used already in step 4; because the copy of e remained unused in membrane r, now l''_i can exit, and it introduces l'_k in the skin region. If e was consumed in step 3, then l''_i can evolve to l'''_i and only now it can leave membrane r, introducing l'_j in the skin region.

The simulation of the SUB instruction is correct. The simulation of ADD and SUB instructions can continue until introducing the object l'_h, which cannot evolve, the computation stops. The contents of membrane 1 correspond to the contents of register 1, hence $N(M) = N_{CPE}(\Pi)$; the observation that we use only symport rules with energy 1 completes the proof. □

The priority of communication is a useful "programming tool"; in its absence, the system has to use more energy quanta.

Theorem 4.2. *$N_{CME}ECP_m(sym_p, anti_q) = NRE$, for all $m \geq 4, p \geq 3, q \geq 0$.*

Proof. Consider a set $Q \in NRE$ and take $Q' = \{n-1 \mid n \in Q\}$. If $1 \notin Q$, then we proceed as follows. We take a register machines with three registers, $M = (3, B, l_0, l_h, R)$, generating the set Q' and we construct the EC P system

$$\Pi = (O, e, [\ [\ \ ]_1 [\ \ ]_2 [\ \ ]_3\ ]_0, l_0', \#, \#, \#, R_0, R_0', R_1, R_1', R_2, R_2', R_3, R_3', 1),$$

with $O = \{l, l', l'', l''', \bar{l} \mid l \in B\} \cup \{a, \#\}$ and the sets of rules as indicated below.

For any ADD instruction of M we use the same rules as in the proof of the previous theorem.

For a SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ in R we consider the following rules:

$$\begin{aligned} R_0 &= \{l_i' \to l_i e,\ l_i''' \to l_j',\ l_i'' \to l_k',\ \# \to \#\}, \\ R_r &= \{l_i \to l_i' e^2,\ l_i' \to l_i'' e,\ l_i'' \to l_i'''\}, \\ R_r' &= \{(l_i e, in),\ (ae^2, out),\ (l_i'' e^3, out),\ (l_i''' e, out),\ (\# e^3, out)\}. \end{aligned}$$

The simulation of the SUB instruction proceeds as follows. Object l_i enters membrane r, produces here two quanta of energy and passes to l_i'. Because of the maximal parallelism, if register r is not empty, in the next step both rules $l_i' \to l_i'' e$ and (ae^2, out) are used, hence we have to pass further to l_i''', which, in the next step, exits to the skin membrane, where it introduces l_j'. If the register r is empty, then no rule can use the two quanta of energy introduced by the rule $l_i \to l_i' e^2$. By using the rule $l_i' \to l_i'' e$, in step 4 one further occurrence of e is introduced in membrane r. In step 5, three rules can be used: $l_i'' \to l_i''', (l_i'' e^3, out)$, and $(\# e^3, out)$; if the first rule is used, then the trap-object $\#$ exits the membrane and the computation never halts, because of the rule $\# \to \#$ from R_0. Thus, the only continuation which can lead to a halting computation is to use the rule $(l_i'' e^3, out)$, and thus l_k' is introduced in the skin region. In both cases, the simulation of the SUB instruction is correct, hence we obtain $N_{CME}(\Pi) = Q$: in membrane 1, besides copies of object a corresponding to the value of register 1 in the halting configuration of M, we also have a copy of $\#$.

If the set Q contains the number 1, then Q' contains 0 instead, which is ignored when considering a register machine for Q'. However, number 1 can be generated separately: consider an additional object, b, present initially in the skin membrane, instead of l_0', and the rules $b \to \lambda$ and $b \to l_0'$. In the first case, the computation stops with only one object in membrane 1, in the latter case we start simulating the computations in the register machine M which generates Q'.

We conclude the proof by observing that the system Π uses symport rules with maximal energy equal to 3. □

Somewhat surprising, the priority of the evolution rules over the communication rules seems to be a weaker feature than using the communication rules with priority, and this can be "explained" by the fact that communication moves objects from a region to another one, hence changes the rules to be applied to the moved objects; in general, the localization (of objects and of rules) is known to be a powerful feature of P systems. As a consequence, a larger number of membranes and amount of energy is needed in this case.

Theorem 4.3. $N_{EPC}ECP_m(sym_p, anti_q) = NRE$, for all $m \geq 7, p \geq 5, q \geq 0$.

Proof. We start again from a register machine $M = (3, B, l_0, l_h, R)$ and construct an EC P system Π simulating it. This time, the components of Π are indicated in a graphical form, in Fig. 1. We explicitly mention the rules associated with an ADD instruction and a SUB instruction operating on register r; specifically, the rules of Π which simulate the ADD instruction are written in the left hand of the figure, under the instruction ADD, and the rules which simulate the SUB instruction are written in the right hand of the figure. The rules associated with the other two registers – denoted in the figure with s and t (hence $\{r, s, t\} = \{1, 2, 3\}$) – are not mentioned.

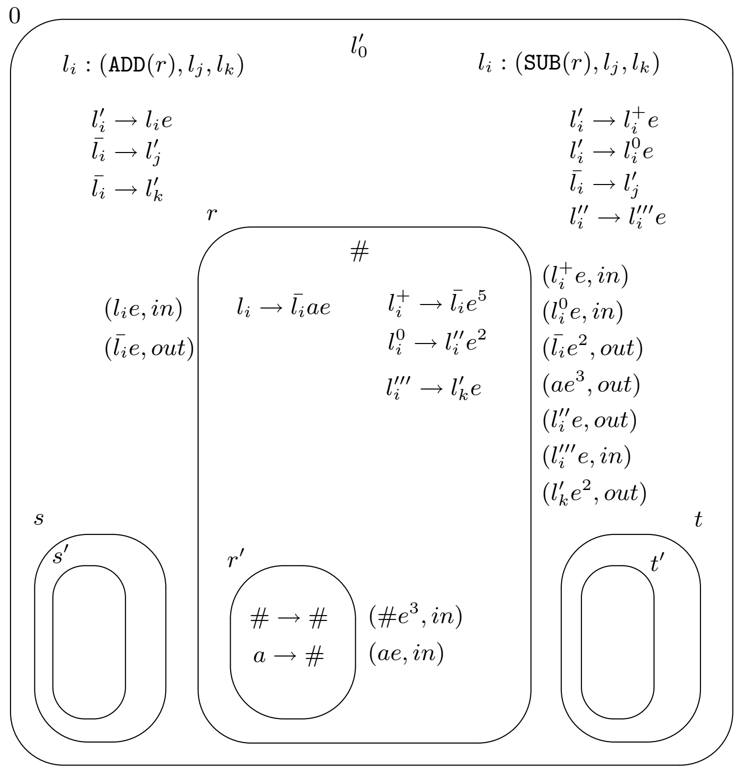

Fig. 1. The P system from the proof of Theorem 4.3.

The simulation of the ADD instruction is obvious, so we only explain the way a SUB instruction is simulated. Object l_i "guesses" whether or not register r is empty, that is it non-deterministically passes to one of l_i^+ and l_i^0. Any of these objects enters membrane r. If l_i^+ was produced and the register was empty, then the computation will enter an infinite cycle. Similarly, if object l_i^0 was produced and the register is not empty the computation will never stop.

Let us assume that we have object l_i^+ in membrane r. It produces here five quanta of energy and passes to $\bar{l}_i$ (remember that l_i uniquely labels an instruction, hence there is no ambiguity in using the bar notation, as in the simulation of the ADD instruction). No evolution rule can be used in the system, hence we are allowed (and we have) to apply communication rules. If any of the rules $(\#e^3, in), (ae, in)$ from $R'_{r'}$ are used, then the trap object # will evolve forever in membrane r'. This can be avoided only if the rules $(\bar{l}_i e^2, out), (ae^3, out)$ of R'_r are used, and this is possible if the register is non-empty; otherwise, no object a is present in membrane r, hence, because of the maximal parallelism, the rule $(\#e^3, in)$ will bring the trap-object in membrane r' and the computation never halts. If $\bar{l}_i$ arrives in the skin region, then it introduces here l'_j, which is the correct continuation of the computation in M.

Assume now that object l_i^0 was produced and introduced in membrane r. By rule $l_i^0 \to l''_i e^2$ we introduce two quanta of energy. One of them can be used by the rule $(l''_i e, out)$ from R'_r. If the membrane contains no copy of a, hence the "guess" made in the first step was correct, i.e., the register is empty, then the rule (ae, in) from $R'_{r'}$ cannot be used, otherwise the computation never stops (because of the maximal parallelism, this rule must be used in parallel with $(l''_i e, out)$; of course, the rule (ae, in) can be used even twice, with the same result, the infinite run of the computation). The copy of e waits inside membrane r until l''_i evolves in the skin region to l'''_i and this object enters membrane r. It produces here one further copy of e and exits in the form of l'_k, thus completing this branch of the simulation.

In both cases, the simulation of the instruction SUB is correct (the computation in Π never ends if an incorrect guess is made or a "wrong" rule is used). The system Π generates the same set of numbers as the register machine M, always augmented by 1: the object # remains in membrane 1 in the end of the computation. Like in the proof of Theorem 4.2, we can now make sure that we generate the set $Q \in NRE$ we want, number 1 included if it belongs to Q, and this completes the proof. $\square$

Several *open problems* can be formulated with respect to these three results, concerning the parameters involved in them: Can the number of membranes be decreased? Can the energy of rules in Theorems 4.2, 4.3 be decreased? Does the use of antiport rules help in this respect? (P systems with minimal symport/antiport rules are already universal – sometimes with some "garbage" objects remaining in the output membrane, see, e.g., the corresponding chapter from [16] – hence the energy can be completely removed if powerful enough communication rules are used.)

5. Dynamical Communication Complexity

Let us now proceed to defining communication complexity parameters starting from computations, not from the (rules of the) system. There are (at least) three basic possibilities: (i) to count the number of steps of a computation when communication operations are done, (ii) to count the total number of communication rules used during a computation, and (iii) to consider the sum of the weights of all communication rules used during a computation (or the energy involved in these rules, in the case of EC P systems with a cost of communication, as considered in the previous sections).

More formally, let $\delta : w_0 \Longrightarrow w_1 \Longrightarrow \ldots \Longrightarrow w_h$ be a halting computation in a given EC P system Π, with w_0 being the initial configuration. We interpret/represent the configurations as strings, specifying the multisets of objects placed in the regions of a membrane structure written as a string of labeled matching parentheses, hence we can speak about the number of occurrences of a symbol in such a string/configuration. Then, for each $i = 0, 1, \ldots, h-1$ we can write:

$$\begin{aligned}
ComN(w_i \Longrightarrow w_{i+1}) &= \begin{cases} 1, & \text{if a communication rule is used in this transition,} \\ 0, & \text{otherwise} \end{cases} \\
ComR(w_i \Longrightarrow w_{i+1}) &= \text{the number of communication rules used} \\
& \text{in this transition,} \\
ComW(w_i \Longrightarrow w_{i+1}) &= \text{the total energy of the communication rules used} \\
& \text{in this transition.}
\end{aligned}$$

These parameters can then be extended in the natural way to computations, results of computations, systems, sets of numbers: for $ComX \in \{ComN, ComR, ComW\}$ we define

$$\begin{aligned}
ComX(\delta) &= \sum_{i=0}^{h-1} ComX(w_i \Longrightarrow w_{i+1}), \text{ for } \delta : w_0 \Longrightarrow w_1 \Longrightarrow \ldots \Longrightarrow w_h \\
& \text{a halting computation,} \\
ComX(n, \Pi) &= \min\{ComX(\delta) \mid \delta : w_0 \Longrightarrow w_1 \Longrightarrow \ldots \Longrightarrow w_h \\
& \text{is a halting computation in } \Pi \text{ with the result } n\}, \\
ComX(\Pi) &= \max\{ComX(n, \Pi) \mid n \in N(\Pi)\}, \\
ComX(Q) &= \min\{ComX(\Pi) \mid Q = N(\Pi)\}.
\end{aligned}$$

In this way, we have the possibility to assess the communication complexity of sets of numbers with respect to EC P systems; actually, the class of systems can be changed, as communication plays an important role in many classes of P systems, so that we can write $ComX_{CL}(Q)$, where CL is a particular class of P systems, and in this way, we can compare the complexity of a set of numbers with respect to various types of systems generating it.

Now, we can also consider families of sets of numbers of a given maximal complexity:

$$NF_{ComX}(k) = \{Q \subseteq \mathbf{N} \mid ComX(Q) \leq k\},$$

for given $k \geq 0$, with $NF_{ComX}(fin)$ being the union of all these families, and $NF_{ComX}(\infty)$ the family of all sets of numbers computed by P systems of a given type. As in most cases, this family is equal to NRE, while non-cooperative P systems without communication generate only semilinear sets, [13], hence we can write

$$\begin{aligned}
&SLIN \subseteq NF_{ComX}(0) \subseteq NF_{ComX}(1) \subseteq \ldots \\
&\ldots \subseteq NF_{ComX}(fin) \subseteq NF_{ComX}(\infty) = NRE.
\end{aligned}$$

Many open problems and research topics arise in this context. First, the previous definition refers to EC P systems (hence to symport/antiport communication rules), but not to the way of using the rules, in the sense of the relation between evolution and communication operations – remember that in the previous sections we distinguished three possibilities in this respect.

This can be very important for the results we obtain: let us consider the following system:

$$\Pi = (\{b, c\}, e, [\,[\quad]_2\,]_1, cbb, \lambda, \{c \rightarrow bbcc, c \rightarrow e\}, \emptyset, \{c \rightarrow c\}, \{(ce, in)\}, 1),$$

and define transitions in the mode CPE, hence with priority of communication over evolution.

Using the rule $c \rightarrow bbcc$, the number of copies of b and c increases exponentially. The number of objects b will always be twice the number of objects c in membrane 1. If in a given step we use both the rule $c \rightarrow bbcc$ and the rule $c \rightarrow e$, then we have at the same time both objects c and e in membrane 1. Hence the rule (ce, in) can be used, with priority. In this case, the computation never stops, since the rule $c \rightarrow c$ can be used forever in membrane 2. Therefore, we either use only the rule $c \rightarrow bbcc$ or only the rule $c \rightarrow e$. After n finite number of steps, the computation will stop with $3 \cdot 2^n$ objects in region 1, for some $n \geq 0$ (the value $n = 0$ is obtained if we use the rule $c \rightarrow e$ in the first step). This means that the system generates a non-semilinear set of numbers.

If we go back to our definition, we can extend it to other classes of P systems. A direct passage is via P systems with active membranes, where we have *in* and *out* rules much similar to the symport rules (but used in a sequential way, one in each membrane). Also transition P systems have communication commands, which can be counted when defining communication complexity parameters as above.

Then, more technically, we ask the following: is the previous hierarchy of complexity classes infinite? We conjecture that for many types of P systems (we believe that this is the case for EC P systems, with or without energy associated with communication rules, without a priority relation between evolution and communication operations) we have the following relations:

$$SLIN = NF_{ComX}(k) = NF_{ComX}(fin) \subset NF_{ComX}(\infty) = NRE,$$

for all $k \geq 0$. Interesting enough intrinsically, such a result would make non-interesting (showing that they are not sensitive enough) the complexity measures themselves. Since only two complexity classes are distinguished, sets of finite complexity and sets of infinite complexity. (This should be contrasted, for instance, with the results related to the index of context-free languages – see details in [6] – which have a somewhat similar definition, but leads to an infinite hierarchy of languages.)

A natural extension is from sets of numbers to sets of vectors of numbers. Sometimes this makes differences between families generated.

6. Communication Complexity as a Computation Regulator

In the previous section, we have considered the "amount of communication" as a property of computations and computing devices. An attractive idea would be to change the perspective and consider (the value of) this parameter as a *condition* to be imposed to computations in a given system. Instead of considering the set of numbers computed by all computations, we consider only a subset of all such numbers which can be obtained in the end of computations with a bounded communication complexity. Specifically, we can define $N_{mode}(\Pi, ComX \leq k)$ to be the set of numbers which can be generated by an EC P system Π (with energy associated with the communication rules) by means of computations δ such that $ComX(\delta) \leq k$, for $k \geq 0$.

In what follows, we consider only the measure $ComN$ (the number of steps when communication rules are used). Like in the previous section, an example is given proving that, at least for modes CPE, CME, systems with a small communication complexity can generate already non-semilinear sets. Consider the system

$$\Pi = (\{a, \#\}, e, [\,[\ ]_2\,]_1, aa\#, \lambda, \{a \to aa, a \to ee\}, \emptyset, \{\# \to \#\}, \{(ae, in), (\#e, in)\}, 2).$$

We have

$$N_{mode}(\Pi, ComN \leq 1) = \{2^n \mid n \geq 1\}, \text{ for } mode \in \{CPE, CME\}.$$

It is clear that the computation can stop only after using the rule $a \to ee$, and without using the rule $(\#e, in)$. If any copy of e is introduced, then a communication must be done, irrespective of whether communication has priority or not (in the CME case, because of the maximal parallelism, the rule $(\#e, in)$ must be used if the rule (ae, in) is not used). This means that after a number $n \geq 0$ of steps, where only the rule $a \to aa$ is used (hence 2^{n+1} copies of a are produced), then we use the rule $a \to ee$.

If we have the same number of copies of a and of e, then all these objects are used by the rule (ae, in), and the computation stops – with only one communication step. The number generated is of the form 2^m, for some $m \geq 1$. If we have more copies of e than of a, then the rule $(\#e, in)$ must be used. In this case, the computation never stops. If we have more copies of a than of e, then either we use the rule $(\#e, in)$ or the rule (ae, in). Using rule $(\#e, in)$ will result to a computation that never ends. While all copies of e will be consumed by the application of the rule (ae, in). The remaining copies of a should evolve further. Thus, in a subsequent step we have to use again the communication rules. This means, the computation will have more than one communication steps, hence it is not accepted. Thus, (halting) computations with at most one communication step generate all and only the powers of 2.

Like before, a more systematic study of using the communication complexity as a tool to regulate computations remains to be done. We could ask several questions which are similar to those formulated in the previous section: Which is the power of finite communication? Does the communication thresholds induce an infinite hierarchy? What about other complexity measures, what about the mode EPC?

7. Communication Complexity of Solving Problems

In this section, let us consider our communication complexity measure as a standard complexity measure in solving problems. As we have already noticed, the measure $ComN$ is nothing else but the parameter time (number of computation steps) when we ignore the evolution steps and counting only the steps when at least one communication rule is used. This implies that the whole range of questions dealt within the theory of time complexity, [18], can be adopted for the new measure. Whether or not anything of interest can be obtained in this way remains to be explored. On the other hand, we could start with what it means to solve a problem Q using a class CL of P systems, and take "for free" the definition of complexity classes with a given communication effort.

In this context, we may start by having a problem Q, characterized by a size parameter *size* taking values from the natural numbers, and the set of instances I_Q taking Boolean values (*true* and *false*). We say that a family $\Pi(n), n \geq 1$, solves (uniformly) the problem Q if (i) each system $\Pi(n)$ is constructed starting from Q and n (hence not from the instances of size n of the problem), and (ii) introducing a code $cod(i_Q)$ of an instance i_Q of size n as a multiset in the skin region of system $\Pi(n)$, the computation halts if and only if i_Q is true (hence the family $\Pi(n), n \geq 1$, is sound and complete with respect to Q).

Note that we have said nothing here about the complexity of solving the problem, that is why we have said nothing about the complexity of constructing the systems $\Pi(n), n \geq 1$, starting from Q and n, neither on the complexity of computing $cod(i_Q)$ starting from i_Q, nor on the complexity of the halting computation in $\Pi(n)$. All these have natural definitions in the case of time complexity, especially when dealing with complexity classes at least polynomial: all these computations should be performed in at most polynomial time.

Here we have a problem: we want also to investigate complexity classes defined according to finite thresholds: such a class contains all problems which can be decided making use of a given number of communication steps, a natural number $k \geq 0$. The construction of the systems $\Pi(n), n \geq 1$, and the computation of $cod(i_Q)$ are done by a Turing machine, and for Turing machines we do not know what communication complexity means.

Therefore, we will ask, as usual questions in time complexity area, to have the systems $\Pi(n), n \geq 1$, constructed by a Turing machine in a polynomial time. The polynomial restriction, however, seems to be too permissive for the computation of the code of the problem instance: in a polynomial time, we can already solve the problem, hence $cod(i_Q)$ can be a single bit, 1 if the instance is true and 0 otherwise. This is not acceptable, and we do not have a general solution to this issue, that is why below we only consider problems whose true instances are described by numerical *relations*.

For instance, for a relation $rel_k \subseteq \mathbf{N}^k$, for some $k \geq 2$, we consider the problem $Qrel_k$ whose instances are of the form $i_{Qrel_k} = (n_1, n_2, \ldots, n_k)$, where $n_i \in \mathbf{N}, 1 \leq i \leq k$, with

$$i_{Qrel_k}(n_1, \ldots, n_k) = \textit{true} \text{ iff } (n_1, n_2, \ldots, n_k) \in rel_k.$$

In such a case, for an instance of size k we input in the skin region a multiset of the form $a_1^{n_1} a_2^{n_2} \dots a_k^{n_k}$, and we expect that the system $\Pi(k)$ halts if and only if the instance has the value *true*.

Another issue appears here, concerning the way the system is working: deterministically, non-deterministically, confluently. In the previous definition, the system is supposed non-deterministic, the instance of the problem is decided to be true if there is a computation which halts, irrespective of how many computations starting in the initial configuration do not halt. Of course, this can be changed, working only with deterministic systems.

In this setup, we say that the problem Q belongs to the complexity class $FComX(s), s \geq 0$, if each instance of size k of Q which is true leads to a halting computation δ of $\Pi(k)$ which has $ComX(\delta) \leq s$, for each $X \in \{N, R, W\}$ (as in Section 5). Note that in Section 5 we have considered families of sets of numbers which can be generated by EC P systems with the communication complexity bounded – more precisely, with *at least* one computation having the communication complexity bounded – while here we consider classes of problems associated with numerical relations, with the relation itself "recognized" by the system by means of computations with a bounded communication complexity. Of course, the classes $FComX(s)$ should also indicate the class of P systems used, but here we always consider EC P systems with quanta of energy, as in the previous sections (in particular, with minimal symport and antiport rules, moving only one object in a direction, with the help/consumption of a quantum of energy).

In what follows, we only briefly investigate the measure $ComN$, i.e., the number of steps when a communication rule is used.

According to the definitions, we have the inclusion $FComN(s) \subseteq FComN(s+1)$, for all $s \geq 0$, and it is expected that these inclusions are strict.

We can prove this for the first inclusion, making use of the problem associated with the equality relation,

$$eq_k = \{(n, n, \dots, n) \mid n \geq 0\} \subset \mathbf{N}^k, \text{ for } k \geq 2.$$

Actually, we believe that this problem can be used in order to prove that the hierarchy $FComN(s), s \geq 0$, is infinite. The constructive part is easy. For instance, in order to show that $Qeq_k \in FComN(k-1)$ we can use the system $\Pi(k)$ given in Fig. 2. This system halts if and only if $n_1 = n_2 = \dots = n_k$, and this is done by means of a computation which has $k-1$ communication steps. In a sequence, one checks whether or not $n_1 = n_2$, $n_1 = n_3$, and so on until $n_1 = n_k$, and each of these subproblems is solved in an easy way: rules associated with a_1 generate continuously copies of e; if their number is equal to the number of copies of $a_i^{(j)}$ produced at the same time, then the computation can continue without producing the trap object $\#$ and without introducing the object d in membrane 2 (in both cases, the computation would continue forever). This means that for each comparison we have a communication step, in total $k-1$, which shows that $Qeq_k \in FComN(k-1)$.

We *conjecture* that $Qeq_k \notin FComN(k-2)$, hence that the hierarchy $FComN(k)$, $k \geq 0$, is infinite.

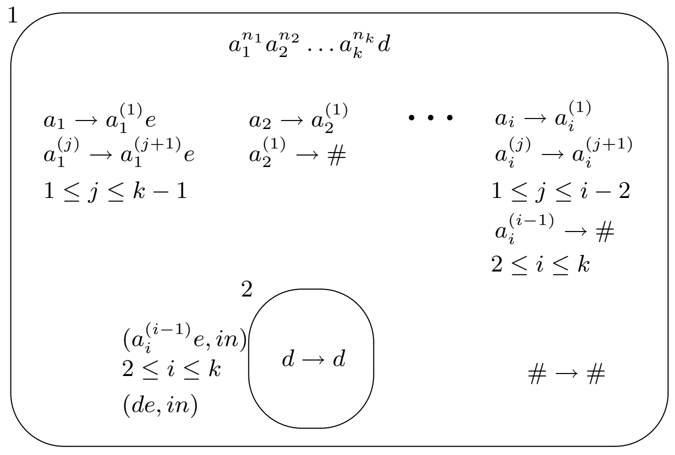

Fig. 2. An EC P system deciding Qeq_k.

A more systematic investigation of classes $FComN(k), k \geq 0$, remains to be carried out, maybe starting with further decision problems based on relations among numbers (hopefully, for one of them the strictness of the hierarchy will be obtained). Which problems can be decided in finite time? Which is the relation between usual complexity classes, with the time/space related by a function to the size of the input problem, and classes defined in a similar way for communication complexity measures?

What about considering other parameters than $ComN$, for instance, $ComR$? In some sense, counting the rules used in each communication step corresponds to the definition of so-called *Sevilla carpet*, see [2], [7]. Can this connection be made more precise?

These and many other questions remain to be answered. Then, all these problems – as well as those formulated in the previous sections – can be extended to other classes of P systems, as communication rules/tools appear in all of them.

8. Closing Remarks

Communication plays an essential role in P systems, so that it is rather natural to investigate this feature more carefully, in particular, to define parameters measuring the communication efforts of a computation, hence of a system. This question is also motivated in view of the fact that P systems are distributed parallel computing devices, hence a classic theory of communication complexity [8] is plausible in this framework. The present paper contributes only preliminarily to this research direction, by considering the effort of communication in EC P systems (in the form of quanta of energy consumed by the communication rules), and by proposing some dynamical communication complexity measures (and some problems and conjectures about them). In some sense, the main aim of this paper was to call the attention to this research area, almost untouched in membrane computing, and we hope that the reader will take this challenge and fill in this gap.

A direct continuation of the present paper is [15], where so-called dP systems are introduced and dP automata are briefly investigated; they are a natural framework for investigating communication complexity issues, thus proposing an answer to the respective question formulated in the present paper. See also [5] for further results about dP systems.

Acknowledgements. The work of of H. Adorna was supported by the Engineering Research and Development for Technology (ERDT) Program of the Department of Science and Technology (DOST), the Philippines. The work of Gh. Păun and M.J. Pérez-Jiménez was supported by Proyecto de Excelencia con Investigador de Reconocida Valía, de la Junta de Andalucía, grant P08 – TIC 04200.

References

[1] Cavaliere M., *Evolution-communication P systems*, Membrane Computing. Proc. WMC 2002, Curtea de Argeş (Gh. Păun et al., eds.), LNCS 2597, Springer, Berlin, 2003, pp. 134–145.

[2] Ciobanu G., Păun Gh., Ştefănescu Gh., *Sevilla carpets associated with P systems*, Proc. Brainstorming Week on Membrane Computing (M. Cavaliere et al., eds.), Tarragona Univ., TR 26/03, 2003, pp. 135–140.

[3] Csuhaj-Varjú E., *P automata*, Membrane Computing. Proc. WMC5, Milano, 2004 (G. Mauri et al., eds.), LNCS 3365, Springer, Berlin, 2005, pp. 19–35.

[4] Csuhaj-Varjú E., Margenstern M., Vaszil G., Verlan S., *On small universal antiport P systems*, Theoretical Computer Sci., 372 (2007), pp. 152–164.

[5] Freund R., Kogler M., Păun Gh., Pérez-Jiménez M.J., *On the power of P and dP automata*, Annals of Bucharest Univ. Mathematical-Informatics Series, 2010, in press.

[6] Gruska J., *Descriptional complexity of context-free languages*, Proc. Symp. on Math. Found. of Computer Science, MFCS, High Tatras, 1973, pp. 71–83.

[7] Gutiérrez-Naranjo M.A., Pérez-Jiménez M.J., Riscos-Núñez A., *Multidimensional Sevilla carpets associated with P systems*, Cellular Computing. Complexity Aspects (M.A. Gutiérrez-Naranjo et al., eds.), Fenix Editora, Sevilla, 2005, pp. 225–236.

[8] Hromkovič J., *Communication Complexity and Parallel Computing: The Application of Communication Complexity in Parallel Computing*, Springer, Berlin, 1997.

[9] Krishna S.N., Păun A., *Some universality results on evolution-communication P systems*, Proc. Brainstorming Week on Membrane Computing; Tarragona, February 2003 (M. Cavaliere et al., eds.), Technical Report 26/03, Rovira i Virgili University, Tarragona, 2003, pp. 207–215.

[10] Leporati A., Zandron C., Mauri G., *Simulating the Fredkin gate with energy-based P systems*, JUCS, **10**, 5 (2004), pp. 600–619.

[11] Leporati A., Mauri G., Zandron C., *Quantum sequential P systems with unit rules and energy assigned to membranes*, Membrane Computing, Proc. WMC6, Vienna, Austria, 2005 (R. Freund et al., eds.), LNCS 3850, Springer, Berlin, 2006, pp. 310–325.

[12] Păun Gh., *Computing with membranes*, Journal of Computer and System Sciences, **61**, 1 (2000), pp. 108–143 (and Turku Center for Computer Science-TUCS Report 208, November 1998, `www.tucs.fi`).

[13] Păun Gh., *Membrane Computing. An Introduction*, Springer, Berlin, 2002.

[14] Păun Gh., *Further open problems in membrane computing*, Proc. Second Brainstorming Week on Membrane Computing, Sevilla, February 2004 (Gh. Păun et al., eds.), Technical Report 01/04 of Research Group on Natural Computing, Sevilla University, Spain, 2004, pp. 354–365.

[15] Păun Gh., Pérez-Jiménez M.J., *Solving problems in a distributed day in membrane computing: dP systems*, Int. J. of Computers, Communication and Control, **5**, 2 (2010), pp. 238–252.

[16] Păun Gh., Rozenberg G., Salomaa A., eds., *Handbook of Membrane Computing*, Oxford University Press, 2010.

[17] Păun Gh., Suzuki Y., Tanaka H., *P systems with energy accounting*, Int. J. Computer Math., **78**, 3 (2001), pp. 343–364.

[18] Pérez-Jiménez M.J., *A computational complexity theory in membrane computing*, Pre-proceedings of WMC10, Curtea de Argeş, Romania, August 2009, Technical Report 01/09 of Research Group on Natural Computing, Sevilla University, Spain, 2009, pp. 82–105.

[19] Porreca A.E., Leporati A., Mauri G., Zandron C., *Introducing a space complexity measure for P systems*, Intern. J. Computers, Communications and Control, **4**, 3 (2009), pp. 301–310.

[20] The P Systems Website: `www.ppage.psystems.eu`