P Systems with String Objects and with Communication by Request

Erzsébet Csuhaj-Varjú, György Vaszil

Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende utca 13–17, H-1111 Budapest, Hungary
{csuhaj,vaszil}@sztaki.hu

Summary. In this paper we study P systems using string-objects where the communication between the regions is indicated by the occurrence of so-called query symbols in the string. We define two variants of communication and prove that these systems with both types of communication are computationally complete, even having a number of membranes limited with relatively small constants.

1 Introduction

In this paper we continue our investigations on P systems with string objects and with communication by request. In [2], the authors studied tissue-like P systems over string objects where the evolution rules of the objects are represented by context-free rewriting rules which also describe the communication between the membranes by the help of communication symbols, called query symbols, one such symbol corresponding to each region of the system.

Membrane systems, or P systems, are distributed and parallel computing devices inspired by the functioning of the living cell [5]. A P system consists of a hierarchically embedded structure of membranes. Each membrane encloses a region that contains objects and might also contain other membranes. There are rules associated to the regions describing the evolution and the movement of the objects which together correspond to a computation. For details on membrane systems, see the monograph [6] and consult the web-page http://psystems.disco.unimib.it.

While in the standard case a P system consists of a hierarchically embedded structure of membranes, tissue-like P systems are organized in another manner [4]. Instead of an individual cell, these correspond to groups of cells, like tissues or organs, interacting with each other either directly or with the use of the environment, but in any case, having the common property that the membrane structures are not necessarily described by a tree as the ones corresponding to individual cells.

Communication in tissue-like P systems with string objects and with communication by request was defined as follows: When one or more query symbols are introduced in a string, then the rewriting of that string stops and the queries are satisfied by replacing the query symbols with strings which do not contain further query symbols from the region indicated by the query symbol, in all possible combinations. If no query symbol free string exists in the queried region, then the string containing the query disappears.

This model has some biological resemblance: if the strings are considered as descriptions of simple organisms, the query symbols as their "weak points", possibly infected or attacked by another organism, then the communication mimics some features of an infection or parasitism. Inspired by these resemblances, we call a communication of type i (infection) if after communicating the copies of the strings, the strings themselves remain in the region, while the communication is called of type p (parasitism), if after communication the communicated string itself disappears from its original region. The model is called an MPC system in short.

MPC systems can also be considered as modified variants of parallel communicating (PC) grammar systems defined over multisets of strings. PC grammar systems are networks of grammars organized in a communicating system to generate a single language. The reader interested in the theory of grammar systems is referred to [1, 7].

In [2], the authors proved that MPC systems with 7 membranes and working with i-communication are able to describe all recursively enumerable languages. The computational completeness of these systems working with p-communication holds as well, even for a subclass consisting of systems having only 9 membranes.

In this paper we define the two types of communication for standard P systems and examine the computational power and the size complexity of these models. We call the new constructs RPC systems in short. In this case, the requested string can only be communicated either to the parent membrane or to one of the child membranes, depending on the issued query symbol. Thus, query symbols refer only to the neighboring regions. According to the above mentioned biological resemblance, both infection and parasitism are very local phenomena regarding their spread, i.e., in one step only the neighbors can be infected and parasitism can be developed only between two closely related, i.e. neighbor components. As for MPC systems, the computational completeness can be proved for RPC systems with both types of communication: in the case of i-communication systems with 10 membranes and in the case of p-communication systems with 30 membranes are enough for demonstrating the power of the Turing machines. The reader can observe that both MPC systems and P systems are able to obtain the computational completeness even with relatively small number of membranes. Moreover, in the case of i-communication the difference between the two numbers is very small, i.e. the difference in the underlying structure of the membrane system has not too much influence on the computational power of the system.

2 Preliminaries and Definitions

We first recall the notions and the notations we use. The reader is assumed to be familiar with the basics of formal language theory, for details see [7]. Let Σ be an alphabet and let Σ^* be the set of all words over Σ, that is, the set of finite strings of symbols from Σ, and let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ where ε denotes the empty word. For $w \in \Sigma^*$ and $S \subseteq \Sigma$, let $|w|_S$ denote the number of occurrences of symbols from S in the string w (if $S = \{a\}$ is a singleton set, we may write $|w|_a$ instead of $|w|_{\{a\}}$).

Let V be a set of objects, and let $\mathbb{N}$ denote the set of non-negative integers. A multiset is a mapping $M : V \to \mathbb{N}$ which assigns to each object $a \in V$ its multiplicity $M(a)$ in M. The support of M is the set $supp(M) = \{a \mid M(a) \geq 1\}$. If $supp(M)$ is a finite set, then M is called a finite multiset. The set of all finite multisets over the set V is denoted by V°.

We say that $a \in M$ if $M(a) \geq 1$. For two multisets $M_1, M_2 : V \to \mathbb{N}$, $M_1 \subseteq M_2$ if for all $a \in V$, $M_1(a) \leq M_2(a)$. The union of M_1 and M_2 is defined as $(M_1 \cup M_2) : V \to \mathbb{N}$ with $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$ for all $a \in V$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2) : V \to \mathbb{N}$ with $(M_1 - M_2)(a) = M_1(a) - M_2(a)$ for all $a \in V$, and the intersection is $(M_1 \cap M_2) : V \to \mathbb{N}$ with $(M_1 \cap M_2)(a) = min(M_1(a), M_2(a))$ for $a \in V$, where $min(x, y)$ denotes the minimum of $x, y \in \mathbb{N}$. We say that M is empty, denoted by ϵ, if its support is empty, $supp(M) = \emptyset$.

In the following we sometimes list elements $a_1, \ldots, a_n$ of a multiset as $M = \{\{a_1, \ldots, a_n\}\}$, by using double brackets to distinguish from the usual set notation.

A P system is a structure of hierarchically embedded membranes, each having a label and enclosing a region containing a multiset of objects and possibly other membranes. The out-most membrane which is unique, is called the skin membrane. The membrane structure is denoted by a sequence of matching parentheses where the matching pairs have the same label as the membranes they represent. If membrane l_i of a given membrane structure μ contains membrane l_j, and there is no other membrane, l_k, such that l_k contains l_j and l_i contains l_k, then we say that membrane l_i is the parent membrane of l_j, denoted as $parent_\mu(l_j) = l_i$, and l_j is one of the child membranes of l_i, denoted as $l_j \in child_\mu(l_i)$. We also define for any region l_i the set of regions $neighbor_\mu(l_i) = \{l_j \mid parent_\mu(l_i) = l_j \text{ or } l_j \in child_\mu(l_i)\}$.

The evolution of the contents of the regions of a P system is described by rules associated to the regions. Applying the rules synchronously in each region, the system performs a computation by passing from one configuration to another one. Several variants of the basic notion have been introduced and studied proving the power of the framework, see the monograph [6] for a summary of notions and results of the area.

In the following we focus on systems where the objects are represented with strings, object evolution is modeled by context-free string rewriting rules, and communication is performed by dynamically emerging requests with the use of query symbols appearing in the string objects.

Definition 1 A string rewriting P system with communication by request or an RPC system (of degree $m \geq 1$) is a construct

$$\Pi = (V, \mu, (M_1, R_1), \ldots, (M_m, R_m), i_o),$$

where:

- $V = N \cup T \cup K$ where N, T, K are pairwise disjoint alphabets of noterminals, terminals, and *query symbols*, respectively, with $K = \{Q_1, \ldots, Q_m\}$ (one query symbol is associated to each region of Π);
- μ is a membrane structure of m membranes;
- $M_1, \ldots, M_m$ are finite multisets over $(N \cup T)^*$;
- $R_1, \ldots, R_m$ are finite sets of context-free rewriting rules of the form $A \to u$, with $A \in N$ and $u \in V^*$;
- $i_o \in \{1, 2, \ldots, m\}$ is the index of the *output* membrane of Π.

The work of such a system starts from the initial configuration $(M_1, \ldots, M_m)$. It passes from a configuration $(M'_1, \ldots, M'_m)$, consisting of multisets of strings over $N \cup T \cup K$ placed in the m regions of the system, to another configuration $(M''_1, \ldots, M''_m)$ in the following way. If no query symbol is present in the strings contained by the system, then each string from each multiset M'_i is rewritten which can be rewritten according to the rules from $R_i, 1 \leq i \leq m$. This means the use of one rule from R_i, non-deterministically chosen, for each string. The strings which cannot be rewritten (no rule can be applied to them) remain unchanged. The resulting multisets of strings are $M''_i, 1 \leq i \leq m$. Note that the rewriting of strings is maximally parallel, in the sense that all strings which can be rewritten must be rewritten, and that the process is non-deterministic, the choice of rules and the places where the rules are applied can lead to several possible new multisets of strings.

If any query symbol is present in any of the strings contained by M'_i, $1 \leq i \leq n$, then a communication is performed: Each symbol Q_j introduced in a string present in region i (that is, in the multiset M'_i), where j is the index of one of the neighboring regions, is replaced with all strings from this neighboring region j which do not contain query symbols. If in region j there are several strings without query symbols, then each of them is used, hence the string from region i is replicated (with the occurrence of Q_j replaced with strings from region j). If there are several query symbols in the same string from component i, then all of them are replaced (we also say that they are *satisfied*) at the same time, in all possible combinations. If a query symbol Q_j cannot be satisfied (region j contains no string without query symbols), then the string containing Q_j is removed (it is like replacing it with the strings from an empty language). We call such a system i-communicating if copies of the requested strings are communicated to the requesting components, and p-communicating if after replacing the query symbols with the requested strings, these strings are removed from the multiset associated to the queried region.

In this way, all query symbols introduced by the rewriting rules disappear, they are either satisfied (replaced by strings without query symbols) or they disappear together with the string which contain them (in the case when they cannot

be satisfied). The multisets obtained in this way in one communication step are $M_1'', \ldots, M_m''$, constituting the next configuration of the system.

We give now the formal definition of the transition.

Definition 2 Let $\Pi = (V, \mu, (M_1, R_1), \ldots, (M_m, R_m), i_o)$ be an RPC system as above, and let $(M_1', \ldots, M_m')$ and $(M_1'', \ldots, M_m'')$ be two configurations of Π. We say that $(M_1', \ldots, M_m')$ directly derives $(M_1'', \ldots, M_m'')$, if one of the following two cases holds.

1. There is no string containing query symbols, that is, $x \in (N \cup T)^*$ for all $x \in \bigcup_{i=1}^m M_i$. In this case, if $M_i' = \{\{x_{i,1}, \ldots, x_{i,t_i}\}\}$, then $M_i'' = \{\{y_{i,1}, \ldots, y_{i,t_i}\}\}$ where either $x_{i,j} \Rightarrow y_{i,j}$ according to a context-free rule of R_i, or $y_{i,j} = x_{i,j}$ if there is no rule in R_i which can be applied to $x_{i,j},\ 1 \leq j \leq t_i,\ 1 \leq i \leq m$.
2. There is at least one $x \in \bigcup_{i=1}^m M_i$ such that $|x|_K > 0$. In this case, rewriting is stopped and a communication step must be performed as follows. Let

$$M_i^{req} = \begin{cases} \{\{x \in M_i' \mid |x|_K = 0\}\} & \text{if there is a } j \in neighbor_\mu(i), \text{ such} \\ & \text{that } y \in M_j' \text{ with } |y|_{Q_i} > 0, \\ \emptyset & \text{otherwise,} \end{cases}$$

let

$$M_i^{avail} = \{\{x \in M_i' \mid |x|_K = 0\}\},$$

for all $i,\ 1 \leq i \leq m$, and let for an $x = x_1 Q_{i_1} x_2 Q_{i_2} \ldots Q_t x_{t+1},\ x_j \in (N \cup T)^*,\ Q_{i_j} \in K,\ 1 \leq j \leq t+1$,

$$Sat(x) = \begin{cases} \{\{x_1 y_{i_1} x_2 y_{i_2} \ldots y_{i_t} x_{t+1} \mid y_{i_j} \in M_{i_j}^{avail}\}\} & \text{if } M_{i_j}^{avail} \neq \emptyset \text{ for} \\ & \text{all } i_j,\ 1 \leq j \leq t, \\ \emptyset & \text{otherwise.} \end{cases}$$

Now, for all $i,\ 1 \leq i \leq m$,

$$M_i'' = M_i' - M_i^{req} - \{\{x \in M_i' \mid |x|_K > 0\}\} + \bigcup_{x \in M_i', |x|_K > 0} Sat(x)$$

in the p-communicating mode, and

$$M_i'' = M_i' - \{\{x \in M_i' \mid |x|_K > 0\}\} + \bigcup_{x \in M_i', |x|_K > 0} Sat(x)$$

in the i-communicating mode.

Let us denote the transitions from one configuration to another, $(M_1', \ldots, M_m')$ to $(M_1'', \ldots, M_m'')$, by $(M_1', \ldots, M_m') \Rightarrow_X (M_1'', \ldots, M_m'')$ with $X = i$ and $X = p$ for i-communicating and p-communicating systems, respectively.

The language generated by the RPC system consists of all terminal strings produced in region i_o during any possible computation in Π.

$$L_X(\Pi) = \{x \in T^* \mid (M_1, \ldots, M_m) \Rightarrow_X^* (M_1', \ldots, M_m') \text{ and } x \in M_{i_o}'\}$$

for $X \in \{i, p\}$, where $\Rightarrow_X^*$ denote the reflexive and transitive closure of $\Rightarrow_X$.

The families of all languages generated in this way by RPC systems of degrees at most $m \geq 1$ with i-communication or p-communication, is denoted by $iRPC_mCF$ and $pRPC_mCF$, respectively. If we use systems of an arbitrary degree, then we replace the subscript m with $*$. Let us also denote the class of recursively enumerable languages by RE.

RPC systems are computationally universal; they characterize the class of recursively enumerable languages, even with a limited number of components.

Before proving this result, we recall the notion of a two-counter machine from [3]. A *two-counter machine* $TCM = (T \cup \{Z, B\}, E, R)$ is a 3-tape Turing machine where T is an *alphabet*, E is a set of *internal states* with two distinct elements $q_0, q_F \in E$, and R is a set of *transition rules*. The machine has a read-only input tape and two semi-infinite storage tapes (the counters). The alphabet of the storage tapes contains only two symbols, Z and B (blank), while the alphabet of the input tape is $T \cup \{B\}$. R contains transition rules of the form $(q, x, c_1, c_2) \to (q', e_1, e_2)$ where $x \in T \cup \{\varepsilon\}$ corresponds to the symbol scanned on the input tape in state $q \in E$, and $c_1, c_2 \in \{Z, *\}$ correspond to the symbols scanned on the storage tapes. If $c_i = Z$, then the symbol scanned on the ith counter tape is Z, if $c_i = *$, then the symbol scanned is either Z or B. By a rule of this form, M enters state $q' \in E$, and the counters are modified according to $e_1, e_2 \in \{-1, 0, +1\}$. If $x \in T$, then the machine was scanning x on the input tape, and the head moves one cell to the right; if $x = \varepsilon$, then the machine performs the transition irrespective of the scanned input symbol, and the reading head does not move.

The symbol Z appears initially on the cells scanned by the storage tape heads and may never appear on any other cell. An integer t can be stored by moving a tape head t cells to the right of Z. A stored number can be incremented or decremented by moving the tape head right or left. The machine is capable of checking whether a stored value is *zero* or not by looking at the symbol scanned by the storage tape heads. If the scanned symbol is Z, then the value stored in the corresponding counter is *zero*. Note that although we do not allow to explicitly check the non-emptiness of the counters which is allowed in [3], this feature can be simulated: After successfully decrementing and incrementing a counter, the stored value is not altered, but the machine can be sure that the scanned symbol is B. A word $w \in T^*$ is accepted by the two counter machine if the input head has read the last non-blank symbol on the input tape, and the machine is in the accepting state q_F. Two-counter machines are computationally complete; they are just as powerful as Turing-machines, see [3].

Theorem 1. $iRPC_{10}CF = RE$.

Proof. We only give the proof of the inclusion $RE \subseteq iRPC_{10}CF$. The reverse inclusion follows from the Church thesis. To this aim, let us consider a recursively enumerable language $L \subseteq T^*$ and a two-counter machine $TCM =$

$(T \cup \{Z, B\}, E, R)$, in the above form, accepting the language L. We construct an RPC system L. Let

$$\Pi = (V, \mu, C_{sel}, C_{gen}, C_{ch_1}, C_{S_4}, C_{c_1}, C_{ind_1}, C_{ch_{2,1}}, C_{c_2}, C_{ind_2}, C_{ch_{2,2}}, sel),$$

where $C_\alpha = (M_\alpha, R_\alpha)$ for $\alpha \in \{sel, gen, c_1, c_2, ind_1, ind_2, ch_1, S_4, ch_{2,1}, ch_{2,2}\}$, $\mu = [\,[\,[\,[\,]_{S_4}\,[\,]_{ind_1}\,[\,]_{ch_{2,1}}]_{c_1}\,[\,[\,]_{ind_2}\,[\,]_{ch_{2,2}}]_{c_2}\,]_{ch_1}\,]_{gen}\,]_{sel}$ and $V = N \cup K \cup T$.

Let $D = \{[q, x, c_1, c_2, q', e_1, e_2] \mid (q, x, c_1, c_2) \to (q', e_1, e_2) \in R\}$, and let us define for any $\alpha = [q, x, c_1, c_2, q', e_1, e_2] \in D$, the following notations: $State(\alpha) = q$, $Read(\alpha) = x$, $NextState(\alpha) = q'$, and $Store(\alpha, i) = c_i$, $Action(\alpha, i) = e_i$ for $i = 1, 2$.

The general idea of the simulation is to represent the states and the transitions of TCM with nonterminals of D and the values of the counters by strings of nonterminals containing as many A symbols as the value stored in the given counter. Let

$N = \{S'_i, F_i, B_i \mid 1 \le i \le 6\} \cup \{S_i, C_i \mid 1 \le i \le 7\} \cup \{\alpha_i, H_i \mid \alpha \in D, 1 \le i \le 8\} \cup \{\alpha'_1, \alpha''_1, \bar{\alpha}'_1, \bar{\alpha}''_1, \underline{\bar{\alpha}}'_1 \mid \alpha \in D\} \cup \{F_{1,i} \mid 1 \le i \le 5\} \cup \{J_i \mid 1 \le i \le 4\} \cup \{A, \bar{F}'_1, \underline{\bar{F}}'_1, \underline{\underline{\bar{F}}}'_1, \bar{F}''_1, E, E_1, I, J, S''_6\}$ and let the rules be defined as follows.

$$\begin{aligned}
M_{sel} &= \{\{I\}\},\\
R_{sel} &= \{I \to \alpha_1 \mid \alpha \in D, State(\alpha) = q_0\} \cup\\
&\quad \{\alpha_8 \to \beta_1 \mid \alpha, \beta \in D, NextState(\alpha) = State(\beta)\} \cup\\
&\quad \{\alpha_8 \to F_1 \mid \alpha \in D, NextState(\alpha) = q_F\} \cup\\
&\quad \{\alpha_i \to \alpha_{i+1} \mid \alpha \in D, 1 \le i \le 7\} \cup \{F_i \to F_{i+1} \mid 1 \le i \le 5\} \cup\\
&\quad \{F_6 \to Q_{gen}, S'_6 \to S'_6, S'_6 \to Q_{gen}, E \to \varepsilon, \bar{\alpha}''_1 \to \varepsilon\}.
\end{aligned}$$

This region keeps track of the current state of the simulated two-counter machine and also selects the transition to be simulated. The symbol I is used to initialize the system by introducing one of the initial transition symbols of the form $[q_0, x, c_1, c_2, q', e_1, e_2]_1$ where q_0 is the initial state. It also produces the result of the computation when after simulating the entering of the counter machine into the final state (that is, after the appearance of the nonterminal F_1), it receives the strings produced in the lower regions and erases the occurrences of the nonterminals E which, if the simulation was successful, produces a terminal word accepted by the two-counter machine.

$$\begin{aligned}
M_{gen} &= \{\{S_1, S'_1\}\},\\
R_{gen} &= \{S_1 \to S_2, S_2 \to Q_{sel}, S'_1 \to Q_{sel}\} \cup\\
&\quad \{\alpha_1 \to \alpha'_1, \alpha_2 \to Q_{sel}, \alpha_i \to \alpha_{i+1} \mid \alpha \in D, 3 \le i \le 6\} \cup\\
&\quad \{\alpha'_1 \to \alpha''_1, \alpha''_1 \to S'_2, \alpha_7 \to xS_1 \mid \alpha \in D, Read(\alpha) = x\} \cup\\
&\quad \{F_1 \to F_{1,1}, F_{1,i} \to F_{1,i+1}, F_{1,5} \to Q_{ch_1} \mid 1 \le i \le 4\} \cup\\
&\quad \{S'_i \to S'_{i+1} \mid 2 \le i \le 4\} \cup \{S'_5 \to S'_1, A \to \varepsilon, J \to \varepsilon\} \cup\\
&\quad \{F_2 \to Q_{sel}, F_i \to F_{i+1} \mid 3 \le i \le 4\} \cup \{F_5 \to Q_{ch_1}\} \cup\\
&\quad \{S'_6 \to S''_6, H_7 \to \varepsilon\}.
\end{aligned}$$

This region generates the string accepted by the counter machine by adding the symbol $Read(\alpha)$ for each $\alpha \in D$ chosen in the selector region. After the appearance of the nonterminal F_1 in the system, this region will append the words from the checking region ch_1 to its own string and send this string which also contains the generated word to the region *sel*. Then it will receive the word from region ch_2 and erase all A and J symbols before forwarding it also to region *sel*.

$$
\begin{aligned}
M_{ch_1} &= \{\{S_1, S_1'\}\},\\
R_{ch_1} &= \{S_1 \to S_2, S_2 \to S_3, S_3 \to Q_{gen}, S_1' \to S_2', S_2' \to Q_{gen}\} \cup\\
&\quad \{\alpha_1'' \to \delta_1\delta_2 Q_{S_4} \mid \alpha \in D, \delta_j = Q_{c_j} \text{ if } Store(\alpha, j) = Z,\\
&\quad\ \text{or } \delta_j = \varepsilon \text{ otherwise}\} \cup \{S_i \to S_{i+1} \mid 4 \le i \le 6\} \cup\\
&\quad \{\alpha_1' \to \bar{\alpha}_1', \bar{\alpha}_1' \to \underline{\bar{\alpha}}_1', \underline{\bar{\alpha}}_1' \to S_3', S_3' \to S_4', S_4' \to S_5'\} \cup\\
&\quad \{F_{1,1} \to \bar{F}_1', \bar{F}_1' \to \underline{\bar{F}}_1', \underline{\bar{F}}_1' \to \underline{\underline{\bar{F}}}_1', \underline{\underline{\bar{F}}}_1' \to Q_{c_1}Q_{c_2}\} \cup\\
&\quad \{S_7 \to S_1, S_5' \to S_1', F_{1,2} \to Q_{S_4}\}, \text{ and}\\
M_{S_4} &= \{\{S_4\}\},\\
R_{S_4} &= \emptyset.
\end{aligned}
$$

The region ch_1 checks whether the counter contents are zero when they should be zero by collecting the counter strings from regions c_1, c_2 when necessary. At the end of the simulation, the collected string is forwarded to the region *gen* and then to region *sel*, where a terminal string can only be produced if the word originating in region ch_1 contains no A symbols.

For $j = 1, 2$, let

$$
\begin{aligned}
M_{c_j} &= \{\{J, C_1\}\},\\
R_{c_j} &= \{J \to J_1, J_1 \to J_2, J_2 \to Q_{ch_1}, A \to Q_{ind_j}, J_3 \to Q_{ind_j}, J_4 \to J\} \cup\\
&\quad \{\bar{\alpha}_1' \to \bar{\alpha}_1'', \bar{\alpha}_1'' \to \delta_\alpha J_3, \underline{\bar{\alpha}}_1' \to C_5 \mid \alpha \in D, \delta_\alpha = A \text{ if } Action(\alpha, j) = 0,\\
&\quad\ \delta_\alpha = AA \text{ if } Action(\alpha, j) = +1, \delta_\alpha = \varepsilon \text{ if } Action(\alpha, j) = -1\} \cup\\
&\quad \{C_i \to C_{i+1}, C_4 \to Q_{ch_1}, C_7 \to C_1 \mid i \in \{1, 2, 3, 5, 6\}\ \} \cup\\
&\quad \{\bar{F}_1' \to \bar{F}_1'', \underline{\bar{F}}_1' \to Q_{ch_{2,j}}, E \to E_1, H_6 \to H_7\}.
\end{aligned}
$$

These regions maintain strings representing the contents of the two counters. After the selection of a transition symbol in the region corresponding to C_{sel}, they execute the action required by the chosen transition symbol by adding AA, A, or ε to the counter string and then deleting one A and J_3 by rewriting it to Q_{ind_j}. The simulation can only be successful if exactly one A and the symbol J_3 is rewritten. This is ensured by region C_{ind_j}. If there is a string obtained after the two queries which contain only a number of A or E symbols and one J_4 symbol, then the simulation of the actions required by the chosen transition was successful. If a counter is empty, this construction also forbids the successful execution of the decrement instruction since this would introduce E_1 in the counter strings.

The rules of the region supporting the work of the counters C_{c_j}, $j = 1, 2$, are defined as follows.

Table 1. Components of Π in the proof of Theorem 1, simulating an instruction with $\alpha = [q, x, Z, B, q', +1, -1]$. The region C_{S_4} is omitted since it always contains the string S_4.

	C_{sel}	C_{gen}	C_{ch_1}	C_{c_1}	C_{ind}	$C_{ch_{2,1}}$
0	β_8	wS_1, S_1'	$E...S_1, S_1'$	$E...J, C_1$	B_1	$AEJ...H_1$
1	α_1	wS_2, Q_{sel}	$E...S_2, S_2'$	$E...J_1, C_2$	B_2	$AEJ...H_2$
1C	α_1	wS_2, α_1	$E...S_2, S_2'$	$E...J_1, C_2$	B_2	$AEJ...H_2$
2	α_2	wQ_{sel}, α_1'	$E...S_3, Q_{gen}$	$E...J_2, C_3$	B_3	$AEJ...H_3$
2C	α_2	$w\alpha_2, \alpha_1'$	$E...S_3, \alpha_1'$	$E...J_2, C_3$	B_3	$AEJ...H_3$
3	α_3	wQ_{sel}, α_1''	$E...Q_{gen}, \bar{\alpha}_1'$	$E...Q_{ch_1}, C_4$	B_4	$AEJ...H_4$
3C	α_3	$w\alpha_3, \alpha_1''$	$E...\alpha_1'', \bar{\alpha}_1'$	$E...\bar{\alpha}_1', C_4$	B_4	$AEJ...H_4$
4	α_4	$w\alpha_4, S_2'$	$E...Q_{c_1}Q_{S_4}, \underline{\bar{\alpha}}_1'$	$E...\bar{\alpha}_1'', Q_{ch_1}$	B_5	$AEJ...H_5$
4C	α_4	$w\alpha_4, S_2'$	$E...\bar{\alpha}_1''S_4, \underline{\bar{\alpha}}_1'$	$E...\bar{\alpha}_1'', \underline{\bar{\alpha}}_1'$	B_5	$AEJ...H_5$
5	α_5	$w\alpha_5, S_3'$	$E...\bar{\alpha}_1''S_5, S_3'$	$E...\delta_\alpha J_3, C_5$	B_6	$AEJ...H_6$
6	α_6	$w\alpha_6, S_4'$	$E...\bar{\alpha}_1''S_6, S_4'$	$E...Q_{ind}J_3, C_6$	E	$AEJ...H_7$
6C	α_6	$w\alpha_6, S_4'$	$E...\bar{\alpha}_1''S_6, S_4'$	$E...EJ_3, C_6$	E	$AEJ...H_7$
7	α_7	$w\alpha_7, S_5'$	$E...\bar{\alpha}_1''S_7, S_5'$	$E...EQ_{ind}, C_7$	J_4	$AEJ...H_8$
7C	α_7	$w\alpha_7, S_5'$	$E...\bar{\alpha}_1''S_7, S_5'$	$E...EJ_4, C_7$	J_4	$AEJ...H_8$
8	α_8	wxS_1, S_1'	$E...\bar{\alpha}_1''S_1, S_1'$	$E...EJ, C_1$	B_1	$AEJQ_{c_1}H_1$
8C	α_8	wxS_1, S_1'	$E...\bar{\alpha}_1''S_1, S_1'$	$E...EJ, C_1$	B_1	$AEJ...H_1$

$$M_{ind_j} = \{\{B_1\}\},$$
$$R_{ind_j} = \{B_i \to B_{i+1} \mid 1 \le i \le 5\} \cup \{B_6 \to E, E \to J_4, J_4 \to B_1\},$$

and

$$M_{ch_{2,j}} = \{\{H_1\}\},$$
$$R_{ch_{2,j}} = \{H_i \to H_{i+1} \mid 1 \le i \le 7\} \cup \{H_8 \to Q_{c_j}H_1\}.$$

Instead of giving a detailed proof of the correctness of our construction, we demonstrate the work of the system in Table 1 and Table 2 by indicating a possible transition sequence of Π while simulating an instruction of the two-counter machine TCM, and by presenting the terminating part of the simulation. Note that the cells of the tables contain only some of the strings produced by the regions, those which are interesting from the point of view of the simulation.

Let us first look at Table 1. The simulated instruction is represented by a nonterminal $\alpha_1 = [q, x, Z, *, q', +1, -1]_1$ chosen in region C_{sel} in the first step. This indicates that the first counter should be empty which requirement is satisfied since region C_{c_1} contains a string containing zero A symbols. In the following few steps, the indexed versions of α reach the regions $C_{gen}, C_{ch_1}, C_{c_j}$, $j \in \{1, 2\}$, and each of these regions executes its part of the simulation. C_{gen} generates the letter read by the two-counter machine, C_{ch_1} queries the regions simulating the counters in the case when their contents should be zero, and this way collects a "checker"

string. If this string contains the nonterminal A, then the simulation is not correct. C_{c_j} maintain the contents of the counters by adding or deleting A-s. Its work is aided by C_{ind} and $C_{ch_{2,j}}$. The region $C_{ch_{2,j}}$ collects the counter strings at the end of each simulating cycle. The simulation was successful if and only if this collected string only contains A, E or J symbols.

The terminating phase of the simulation is presented on Table 2. When G_{sel} selects the symbol F, the system prepares to finish its work. The variously indexed versions of F travel through the system and result in the transfer of the word generated in G_{gen} and the checker string of region C_{ch_1} to region C_{sel}. There the symbol A cannot be erased, so a terminal word can only be produced if the checker string does not contain this symbol. Meanwhile, the other checker strings are transferred from $C_{ch_{2,j}}$ to region C_{gen} where the A and J symbols can be erased, but nothing else, so when later also this string is transferred to C_{sel}, a terminal string can only be produced if the behavior of the counter simulating regions were correct in each step of the simulation. The last row of the table represents the situation when the erasing process begins. When all A and J have disappeared from the string in region C_{gen}, then S'_6 can be changed to a query symbol transferring the result to C_{sel}, where all remaining symbols can be erased, in the case when the simulation was correct. □

Next we prove that any RPC system using i-communication can be simulated with an RPC system using p-communication.

Theorem 2. *$iRPC_nCF \subseteq pRPC_{3n}CF$, for any $n \geq 1$.*

Proof. Let $\Pi = (V, \mu, (M_1, R_1), \ldots, (M_n, R_n), 1)$ be a system of degree n with $V = N \cup K \cup T$. We construct $\Pi' = (V', \mu', (M'_1, R'_1), \ldots, (M'_{3n}, R'_{3n}), 1)$ of degree $3n$, such that $L_p(\Pi') = L_i(\Pi)$.

Let μ' be defined by adding two new regions $[\,[\,]_{2n+i}\,]_{n+i}$ inside every region i. This way, $n + i \in neighbor_{\mu'}(i)$, and $2n + i \in neighbor_{\mu'}(n + i)$ for all $1 \leq i \leq n$.

Let $V' = N' \cup K' \cup T$ where $N' = N \cup \{S_1, S_2, S_3, S_4\}$, and let the rules of Π' be defined as

$$M'_i = M_i \cup \{\{S_1, S_2\}\},$$
$$R'_i = R_i \cup \{S_1 \to Q_i, S_2 \to Q_{n+i}\},$$

and

$$M'_{n+i} = \{\{S_1, S_2, S_3, S_4\}\}, \quad R'_{n+i} = \{S_3 \to Q_{n+i}, S_4 \to Q_{2n+i}\},$$
$$M'_{2n+i} = \{\{S_1, S_2, S_3, S_4\}\}, R'_{2n+i} = \{S_1 \to Q_{n+i}, S_2 \to Q_{2n+i}\}$$

for all $1 \leq i \leq n$.

The additional membranes of Π' work as "suppliers" of symbols. In each step, each region i rewrites S_1 and S_2 to query itself, and the region $n+i$. From itself it "receives" the strings it contains besides S_1, S_2, from region $n+i$ it receives S_1, S_2, so the same behavior can be repeated in the next step. This self query mechanism

Table 2. Components of Π in the proof of Theorem 1, simulating the terminating phase of the system. The region C_{S_4} is omitted since it always contains the string S_4.

	C_{sel}	C_{gen}	C_{ch_1}	C_{c_1}	C_{ind}	$C_{ch_{2,1}}$
0	β_8	wS_1, S_1'	$E...S_1, S_1'$	$E...J, C_1$	B_1	$AEJ...H_1$
1	F_1	wS_2, Q_{sel}	$E...S_2, S_2'$	$E...J_1, C_2$	B_2	$AEJ...H_2$
1C	F_1	wS_2, F_1	$E...S_2, S_2'$	$E...J_1, C_2$	B_2	$AEJ...H_2$
2	F_2	$wQ_{sel}, F_{1,1}$	$E...S_3, Q_{gen}$	$E...J_2, C_3$	B_3	$AEJ...H_3$
2C	F_2	$wF_2, F_{1,1}$	$E...S_3, F_{1,1}$	$E...J_2, C_3$	B_3	$AEJ...H_3$
3	F_3	$wQ_{sel}, F_{1,2}$	$E...Q_{gen}, \bar{F}_1'$	$E...Q_{ch_1}, C_4$	B_4	$AEJ...H_4$
3C	F_3	$wF_3, F_{1,2}$	$E...F_{1,2}, \bar{F}_1'$	$E...\bar{F}_1', C_4$	B_4	$AEJ...H_4$
4	F_4	$wF_4, F_{1,3}$	$E...Q_{S_4}, \underline{\bar{F}}_1'$	$E...\bar{F}_1'', Q_{ch_1}$	B_5	$AEJ...H_5$
4C	F_4	$wF_4, F_{1,3}$	$E...S_4, \underline{\bar{F}}_1'$	$E...\bar{F}_1'', \underline{\bar{F}}_1'$	B_5	$AEJ...H_5$
5	F_5	$wF_5, F_{1,4}$	$E...S_5, \underline{\underline{\bar{F}}}_1'$	$E...\bar{F}_1''$ $Q_{ch_{2,1}}$	B_6	$AEJ...H_6$
5C	F_5	$wF_5, F_{1,4}$	$E...S_5, \underline{\underline{\bar{F}}}_1'$	$E...\bar{F}_1''$ $AEJ...H_6$	B_6	$AEJ...H_6$
6	F_6	wQ_{ch_1} $F_{1,5}$	$E...S_6$ $Q_{c_1}Q_{c_2}$	$E...\bar{F}_1''$ $AEJ...H_7$	E	$AEJ...H_7$
6C	F_6	$wE...S_6$ $F_{1,5}$	$E...S_6$ $AEJ.....H_7$	$E...\bar{F}_1''$ $AEJ...H_7$	E E	$AEJ...H_7$ $AEJ...H_7$
7	Q_{gen}	$wE...S_6'$ Q_{ch_1}	$E...S_7$ $AEJ.....H_7$	$E...\bar{F}_1''$ $AEJ...H_7$	J_4	$AEJ...H_8$
7C	$wE...S_6'$	$wE...S_6'$ $AEJ...H_7$	$E...S_7$ $AEJ.....H_7$	$E...\bar{F}_1''$ $AEJ...H_7$	J_4	$AEJ...H_8$

is used in each region to keep a copy of its contents even in the case when it is requested by some other region. This way, Π' simulates the communication behavior of Π. □

By Theorems 1 and 2, we obtain the immediate corollary.

Corollary 3 $iRPC_{10} = pRPC_{30}CF = RE$.

3 Closing Remarks

We proved that in the case of string rewriting P systems communication according to dynamically emerging requests leads to computational completeness both for standard P systems and tissue-like P systems, even in the case of systems with bounded size parameters. There have remained several open problems for further study. For example, it is not known whether the obtained size bounds are sharp or not, and whether or not the sharp bounds are different for MPC systems and RPC systems.

References

1. E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.
2. E. Csuhaj-Varjú, Gh. Păun, Gy. Vaszil, Tissue-like P systems communicating by request. Submitted.
3. P. C. Fischer, Turing machines with restricted memory access, *Information and Control*, 9 (1966), 364–379.
4. C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodríguez-Patón, Tissue P systems, *Theoretical Computer Science*, 296(2) (2003), 295–326.
5. Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences* 61(1) (2000), 108–143 (and Turku Center for Computer Science-TUCS Report 208, November 1998, `www.tucs.fi`).
6. Gh. Păun, *Membrane Computing: An Introduction*, Springer-Verlag, Berlin, 2002.
7. G. Rozenberg, A. Salomaa, (eds.), *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.