Computationally Complete Generalized Communicating P Systems with Three Cells

Erzsébet Csuhaj-Varjú[1] and Sergey Verlan[2]

[1] Department of Algorithms and Their Applications,
Faculty of Informatics,
ELTE Eötvös Loránd University, Budapest,
Pázmány Péter sétány 1/c, 1117, Hungary
csuhaj@inf.elte.hu
[2] Université Paris Est, LACL (EA 4219), UPEC, F-94010, Créteil, France
verlan@u-pec.fr

Abstract. Generalized communicating P systems are particular variants of networks of cells where each rule moves only two objects. In this paper we show that GCPSs with three cells and with only join, or only split, or only chain rules are computationally complete computing devices. These bounds are improvements of the previous results.

1 Introduction

Purely communicating P systems are of particular interest in membrane computing [9]. These membrane systems work without any change of their objects but only with importing/exporting objects from and/or to the environment and communicating objects between their regions. A lot of these P system variants are computationally complete, demonstrating that rewriting can be replaced by communication with the environment where some objects are supposed to be found in an unbounded number of copies. This means that whenever to complete a transition the P system needs more (a finite number of new) objects than it has inside, then these objects are always available.

Generalized communicating P systems, introduced in [12] are such models, originally with the aim of providing a common generalization of various purely communicating models.

A generalized communicating P system, or a GCPS for short, is a tissue-like P system where each node represents a cell and each edge is represented by a rule. Every node contains a multiset of objects that can be communicated, i.e., it may move between the cells according to interaction (communication) rules.

The form of an interaction rule is $(a,i)(b,j) \to (a,k)(b,l)$ where a and b are objects and i,j,k,l are labels identifying the input and the output cells. Such a rule means that an object a from cell i and an object b from cell j move synchronously (in one step) to cell k and cell l, respectively. These rules are particularly simple, since they describe the move of only two objects.

The system is embedded in an environment, represented by cell 0, which may have certain objects in an infinite number of copies and certain objects

only in a finite number of copies. The generalized communicating P system and the environment interact by using the communication (interaction) rules given above, with the restriction that at every computation step only a finite number of objects is allowed to enter in any cell from the environment.

The rules are applied in a maximally parallel manner, possibly changing the multisets representing the contents of the cells (the configuration of the GCPS). A computation in a GCPS is a sequence of configurations directly following each other, starting from the initial configuration and ending in a halting configuration. The result of the computation is the number of objects found in a distinguished cell, the output cell.

Due to their simplicity and relation to other fields like the theory of Petri nets, GCPSs have been studied in details. It has been shown that even restricted variants of these constructs (with respect to the form of rules) are able to generate any recursively enumerable set of numbers. Furthermore, several of them even with relatively small numbers of cells and with simple underlying hypergraph architectures are computationally complete [3], [5–7]. It is also shown that the maximal expressive power can also be obtained with GCPSs where the alphabet of objects is a singleton [2]. Furthermore, computational completeness with small number of cells can also be obtained if the objects of the environment are provided step by step with a multiset generating system [1].

In the paper, we demonstrate that computational completeness of three restricted variants, namely, GCPSs with only three cells and with only join rules, or only split rules, or only chain rules are computationally complete. The proofs are based on simulating the register machines [8] and by using the formal framework of P systems [4].

2 Basic Notions

The reader is supposed to be familiar with formal language theory and membrane computing; for further details consult [10] and [9].

For a finite multiset of symbols M over an alphabet V, $supp(M)$ denotes the set of symbols in M (the support of M) and $|M|$ denotes the total number of its symbols (its size). The number of occurrences of symbol x in M is denoted by $|M|_x$. The set of all finite multisets over V is denoted by V°.

Throughout the paper, every finite multiset M is presented as a string w, where M and w have the same number of occurrences of symbol a, for each $a \in V$. The empty multiset is denoted by λ.

A register machine [8] is a 5-tuple $M = (Q, R, q_0, q_f, P)$, where Q is a finite non-empty set, called the set of states, $R = \{A_1, \ldots, A_k\}$, $k \geq 1$, is a set of registers, $q_0 \in Q$ is the initial state, and $q_f \in Q$ is the final state. P is a set of instructions of the following forms: $(p, A+, q, s)$, where $p, q, s \in Q, p \neq q_f, A \in R$, called an increment instruction, or $(p, A-, q, s)$, where $p, q, s \in Q, p \neq q_f, A \in R$, called a decrement instruction. For every $p \in Q$, $(p \neq q_f)$, there is exactly one instruction of the form either $(p, A+, q, s)$ or $(p, A-, q, s)$.

A configuration of a register machine M, defined above, is a $(k+1)$-tuple $(q, m_1, \ldots, m_k)$, where $q \in Q$ and $m_1, \ldots, m_k$ are non-negative integers; q is the current state of M and $m_1, \ldots, m_k$ are the current numbers stored in the registers (the current contents of the registers or the value of the registers) $A_1, \ldots A_k$, respectively.

A transition of the register machine consists in executing an instruction. An increment instruction $(p, A+, q, s) \in P$ is performed if M is in state p, the number stored in register A is increased by 1, and after that M enters either state q or state s, chosen non-deterministically. A decrement instruction $(p, A-, q, s) \in P$ is performed if M is in state p, and if the number stored in register A is positive, then it is decreased by 1, and then M enters state q, and if the number stored in A is 0, then the contents of A remains unchanged and M enters state s.

A register machine $M = (Q, R, q_0, q_f, P)$, with k registers, given as above, generates a non-negative integer n if starting from the initial configuration $(q_0, 0, 0, \ldots, 0)$ it enters the final configuration $(q_f, n, 0, \ldots, 0)$. The set of non-negative integers generated by M is denoted by $N(M)$.

Next we recall the basic definitions concerning generalized communicating P systems [12].

A generalized communicating P system (a GCPS) of degree n, where $n \geq 1$, is an $(n+4)$-tuple $\Pi = (O, E, w_1, \ldots, w_n, R, h)$ where

1. O is an alphabet, called the set of objects of Π;
2. $E \subseteq O$; called the set of environmental objects of Π;
3. $w_i \in O^*$, $1 \leq i \leq n$, is the multiset of objects initially associated to cell i;
4. R is a finite set of interaction rules or communication rules of the form $(a, i)(b, j) \to (a, k)(b, l)$, where $a, b \in O$, $0 \leq i, j, k, l \leq n$, and if $i = 0$ and $j = 0$, then $\{a, b\} \cap (O \setminus E) \neq \emptyset$; i.e., $a \notin E$ and/or $b \notin E$;
5. $h \in \{1, \ldots, n\}$ is the output cell.

The system consists of n cells, labeled by natural numbers from 1 to n, which contain multisets of objects over O. Initially, cell i contains multiset w_i (the initial contents of cell i is w_i). An additional special cell, labeled by 0 and called the environment is distinguished. The environment contains objects of E in an infinite number of copies.

The cells interact by means of the rules $(a, i)(b, j) \to (a, k)(b, l)$, with $a, b \in O$ and $0 \leq i, j, k, l \leq n$. As the result of the application of the rule, object a moves from cell i to cell k and b moves from cell j to cell l. If two objects from the environment move to some other cell or cells, then at least one of them must not appear in the environment in an infinite number of copies.

A configuration of a GCPS Π, as above, is an $(n+1)$-tuple $(z_0, z_1, \ldots, z_n)$ with $z_0 \in (O \setminus E)^*$ and $z_i \in O^*$, for all $1 \leq i \leq n$; z_0 is the multiset of objects present in the environment in a finite number of copies, whereas, for all $1 \leq i \leq n$, z_i is the multiset of objects present inside cell i. The initial configuration of Π is the $(n+1)$-tuple $(\lambda, w_1, \ldots, w_n)$.

Given a multiset of rules $\mathcal{R}$ over R and a configuration $u = (z_0, z_1, \ldots, z_n)$ of Π, we say that R is applicable to u if all its elements can be applied simultaneously to the objects of multisets $z_0, z_1, \ldots, z_n$ such that every object is used

by at most one rule. Then, for a configuration $u = (z_0, z_1, \ldots, z_n)$ of Π, a new configuration $u' = (z'_0, z'_1, \ldots, z'_n)$ is obtained by applying the rules of R in a non-deterministic maximally parallel manner.

One such application of a multiset of rules satisfying the conditions listed above represents a transition in Π from configuration u to configuration u'. A transition sequence is said to be a successful generation by Π if it starts with the initial configuration of Π and ends with a halting configurations, i.e., with a configuration where no further transition step can be performed.

Π generates a non-negative integer n if there is a successful generation by Π such that n is the size of the multiset of objects present inside the output cell in the halting configuration. The set of non-negative integers generated by a GCPS Π in this way is denoted by $N(\Pi)$.

In the following we recall the notions of the possible restrictions on the interaction rules (modulo symmetry). We distinguish the following cases, called GCPSs with minimal interaction:

1. $i = j = k \neq l$: the conditional-uniport-out rule (the *uout* rule) sends b to cell l provided that a and b are in cell i [11];
2. $i = k = l \neq j$: the conditional-uniport-in rule (the *uin* rule) brings b to cell i provided that a is in that cell;
3. $i = j$, $k = l$, $i \neq k$: the symport2 rule (the $sym2$ rule) corresponds to the minimal symport rule [9], i.e., a and b move together from cell i to k;
4. $i = l$, $j = k$, $i \neq j$: the antiport1 rule (the $anti1$ rule) corresponds to the minimal antiport rule [9], i.e., a and b are exchanged in cells i and k;
5. $i = k$ and $i \neq j$, $i \neq l$, $j \neq l$: the presence-move rule (the *presence* rule) moves the object b from cell j to l, provided that there is an object a in cell i and i, j, l are pairwise different cells;
6. $i = j$, $i \neq k$, $i \neq l$, $k \neq l$: the *split* rule sends a and b from cell i to cells k and l, respectively;
7. $k = l$, $i \neq j$, $k \neq i$, $k \neq j$: the $join$ rule brings a and b together to cell k;
8. $l = i$, $i \neq j$, $i \neq k$ and $j \neq k$: the *chain* rule moves a from cell i to cell k while b is moved from cell j to cell i, i.e., to the cell where a located previously;
9. i, j, k, l are pairwise different numbers: the parallel-shift rule (the $shift$ rule) moves a and b from two different cells to another two different cells.

$NOtP_k(x)$ denotes the set of numbers generated by generalized communicating P systems with minimal interaction of degree k, $k \geq 1$, and with rules of type x, where $x \in \{uout, uin, sym2, anti1, presence, split, join, chain, shift\}$. $NOtP_*(x)$ is the notation for $\bigcup_{k=1}^{\infty} NOtP_k(x)$.

Generalized communicating P systems are particular variants of network of cells, constructs introduced in [4] as a formal framework of P systems.

A network of cells of degree $n \geq 1$ is a construct $\Pi = (n, O, w, Inf, R)$ where

- n is the number of cells;
- O is an alphabet;
- $w = (w_1, \ldots, w_n)$ where $w_i \in O^\circ$, for all $1 \leq i \leq n$, is the finite multiset initially associated to cell i;

- $Inf = (Inf_1, \ldots, Inf_n)$ where $Inf_i \subseteq O$, for all $1 \leq i \leq n$, is the set of symbols which may occur in infinitely many copies in cell i (in most of the cases, only one cell, called the environment, may contain symbols with infinite multiplicity);
- R is a finite set of rules of the form $(X \to Y; P, Q)$ where $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$, $x_i, y_i \in V^\circ$, $1 \leq i \leq n$, are vectors of multisets over O and $P = (p_1, \ldots, p_n)$, $Q = (q_1, \ldots, q_n)$, p_i, q_i, $1 \leq i \leq n$ are finite sets of multisets over O. We will also use the notation

$$(1, x_1) \ldots (n, x_n) \to (1, y_1) \ldots (n, y_n) \, ; [(1, p_1) \ldots (1, p_n)]; [(1, q_1) \ldots (n, q_n)]$$

  for a rule $(X \to Y; P, Q)$; moreover, if some p_i or q_i is an empty set or some x_i or y_i is equal to the empty multiset, $1 \leq i \leq n$, then we may omit it from the specification of the rule.

The above rule means the following: objects x_i from cells i are rewritten into objects y_j in cells j, $1 \leq i, j \leq n$, if every cell k, $1 \leq k \leq n$, contains all multisets from p_k and does not contain any multiset from q_k.

A configuration C of Π is an n-tuple of multisets $(u_1, \ldots, u_n)$ over O where $u_i \cap Inf_i = \emptyset$, $1 \leq i \leq n$.

Networks of cells compute sets of numbers; the result of the computation can be defined in several manners, among other by the number of objects in a distinguished cell in a halting configuration.

It is easy to see that GCPSs are particular variants of networks of cells: any rule $(a, i)(b, j) \to (a, k)(b, l)$ of a generalized communicating P system corresponds to a rule $(i, a)(j, b) \to (k, a)(l, b)$ in the corresponding network of cells. Obviously, if the GCPS is with minimal interaction, the form of the rules in the corresponding network of cells is modified accordingly.

Thus, without any proof, we may state that for any generalized communicating P system $\Pi = (O, E, w_1, \ldots, w_n, R, h)$, $1 \leq h \leq n$, there exists a network of cells $\Pi' = (n, O, w, Inf, R)$ of degree n such that $N(\Pi) = N(\Pi')$ and Π and Π' strongly simulate each other. (In the case of a strong simulation, one step of the simulated system is performed using one step in the simulating system. If two systems can simulate each other, then we speak about bi-simulation.)

3 Main Results

In the following we present the computational completeness results concerning generalized communicating P systems with minimal interaction. For simplicity, throughout the paper we follow the notations used for networks of cells.

In [3] it was shown that GCPSs with 7 cells and only join rules are computationally complete. The result was improved in [5, 6] to bound 4. Here we present a further improvement.

Theorem 1. $NOtP_3(join) = NRE$.

Proof. Let us consider an arbitrary register machine $M = (Q, R, q_0, q_f, P)$ with $R = \{A_1, \dots A_n\}$, $n \geq 1$, given as in Section 2. To prove the statement, we construct a generalized communicating P system $\Pi = (O, E, w_1, w_2, w_3, R_1, 2)$ with join rules such that $N(\Pi) = N(M)$. The proof is based on the simulation of M by Π and conversely, i.e., by showing that for any successful generation in M there exists a successful generation in Π and conversely such that the two generation processes yield the same number as result.

Since for every $p \in Q$, $(p \neq q_f)$, there is exactly one instruction of the form either $(p, A+, q, s)$ or $(p, A-, q, s)$, the set of instructions R of M can be labeled by the elements of Q in a one-to-one manner.

Let Q^+ and Q^- be the sets of labels of the increment instructions and the decrement instructions of M, respectively.

Let us define the alphabet of objects of Π as
$O = Q \cup R \cup \{p' \mid p \in Q\} \cup \{\bar{p}, p_1 \mid p \in Q^-\} \cup \{C_i \mid A_i \in R\}$.

Let $E = Q \cup R \cup \{p' \mid p \in Q^+\} \cup \{p_1 \mid p \in Q^-\} \cup \{C_i \mid A_i \in R\}$. and $w_1 = \{q_0\}$, $w_2 = \emptyset$, $w_3 = \{\bar{p} \mid p \in Q^-\}$.

The set of rules R_1 of Π is defined as follows.

For any instruction (p, A_i+, q, s) of M we add the following rules to R_1:

$$p.1 : (1, p)(0, q') \to (3, pq') \qquad p.1' : (1, p)(0, s') \to (3, ps')$$
$$p.2 : (3, p)(0, A_i) \to (2, pA_i)$$

For any instruction (p, A_i-, q, s) of M we add the following rules to R_1:

$$p.1 : (1, p)(0, C_i) \to (3, pC_i) \qquad p.2 : (3, p)(0, p_1) \to (2, pp_1)$$
$$p.3 : (2, p_1)(1, \bar{p}) \to (0, p_1\bar{p}) \qquad p.4 : (0, \bar{p})(3, C_i) \to (1, \bar{p}C_i)$$
$$p.5 : (0, \bar{p})(1, C_i) \to (2, \bar{p}C_i) \qquad p.6 : (2, \bar{p})(0, q') \to (3, \bar{p}q')$$
$$p.7 : (1, \bar{p})(0, s'') \to (2, \bar{p}s'') \qquad p.8 : (3, s'')(1, C_i) \to (2, s''C_i)$$
$$p.9 : (2, s'')(0, s') \to (3, s''s') \qquad p.10 : (3, s'')(2, C_i) \to (0, s''C_i)$$

We also add following rules to R_1:

$$p.p' : (3, p')(0, p) \to (1, p'p), \text{ for all } p \in Q$$
$$p.q : (2, p)(1, q') \to (0, pq'), \text{ for all } p, q \in Q$$
$$c_i.1 : (3, C_i)(2, A_i) \to (1, C_iA)$$
$$c_i.2 : (2, C_i)(1, A_i) \to (0, C_iA_i)$$

Now we prove that Π simulates M. For this, we show how the rules given above simulate the instructions of M. We first note that the simulation of any instruction of M starts with a symbol p in cell 1 which the corresponds to a state of M and no other symbol corresponding to a state of M can be found in this cell. However, cell 1 contains symbol $\bar{p}$ for any state p of M which appears in a decrement instruction. During the simulation of any instruction of M, Π cannot start the simulation of some other instruction, thus the simulating phases do not

interfere. The contents of register A_i is represented by the number of symbols A_i appearing in the cells, symbol C_i assists the simulation of the decrement of register A_i, $1 \leq i \leq n$.

We start with the simulation of instructions of the form (p, A_i+, q, s). After applying rule $p.1$ or $p.1'$, respectively, symbols q' or s' move to cell 3. Then by applying rules $p.2$ and $p.p'$ in parallel, cell 2 will contain one more symbol A_i. In the next step, by applying rules $q.q'$ or $s.s'$ and $p.q$ or $p.s$, respectively, in parallel, symbol q or s enters cell 1 and q' or s' leave the system. Thus, the simulation of (p, A_i+, q, s) has completed.

The simulation of instruction (p, A_i-, q, s) is as follows: First rule $p.1$ is applied and thus p and C_i enter cell 3. Then, depending on whether or not cell 2 contains at least one copy of A_i (register A_i is empty or not) the following rules are applied. If cell 2 contains at least one A_i, then by rule $c_i.1$ symbols C_i and A_i move to cell 1. Meantime, by applying rule $p.2$, and then $p.3$ and $p.5$, $\bar{p}$ enters cell 2 and C_i moves from cell 1 to cell 2. Then, by rule $c_i.2$, symbols A_i and C_i leave the system and by rule $p.6$ symbol $\bar{p}$ introduces a copy of q' in cell 3. Then, by performing rule $q.q'$ and $p.q$, symbol q arrives in cell 1, and the simulation of the next instruction (if $q \neq q_f$) may start. If cell 2 does not contain any copy of A_i, then C_i in cell 3 introduces from the environment the copy of $\bar{p}$ that was sent out the system before. Then by rules $p.7$ and $p.9$ symbols s'' and s' move to cell 3. After then, by executing rule $p.8$ and $p.10$, symbols C_i and s'' leave the system, meantime by rules $s.s'$ and $p.s$ symbol s moves to cell 1, and thus the simulation of the instruction ends.

The reader may notice that the rules can be performed only in the manner described above. This implies that any computation in M can correctly be simulated by Π and $N(M) = N(\Pi)$ holds. Since $N(M)$ is a recursively enumerable set of numbers, the statement of the theorem holds.

Next we show that three cells (and the environment) are sufficient to obtain computational completeness in case of GCPSs with only split rules. In [3] it was shown that GCPSs with 9 cells and only split rules are computationally complete, in [5, 6] the bound was improved to 5.

Theorem 2. $NOtP_3(split) = NRE$.

Proof. Let us consider an arbitrary register machine $M = (Q, R, q_0, q_f, P)$ with $R = \{A_1, \ldots A_n\}$, $n \geq 1$, given as in Section 2. To prove the statement, we construct a generalized communicating P system $\Pi = (O, E, w_1, w_2, w_3, R_1, 2)$ with split rules such that $N(\Pi) = N(M)$. The proof is based on the simulation of M by Π and vice versa, i.e., by showing that for any successful generation in M there exists a successful generation in Π and conversely such that the two generation processes yield the same number as result.

Let Q^+ and Q^- be the sets of labels of the increment instructions and the decrement instructions of M, respectively.

Let us define the alphabet of objects of Π as
$O = Q \cup R \cup \{p' \mid p \in Q\} \cup \{\bar{p}, p_1, p_2, p_3 \mid p \in Q^-\} \cup \{S_i \mid A_i \in R\} \cup \{Z, Z', Z''\}$.

Let $E = Q \cup R \cup \{Z''\}$ and $w_1 = \{q_0\} \cup \{S_i \mid A_i \in R\} \cup \{p_1 \mid p \in Q^-\}$, $w_2 = \{Z\}$, $w_3 = \{Z'\} \cup \{q' \mid q \in Q\} \cup \{p_2, p_3 \mid p \in Q^-\}$.

The set of rules R_1 of Π is defined as follows.

For any instruction (p, A_i+, q, s) of M we add the following rules to R_1:

$$p.1 : (1, pS_i) \to (0, S_i)(3, p) \qquad p.2 : (0, S_iA_i) \to (1, S_i)(2, A_i)$$
$$p.3 : (3, pq') \to (2, p)(0, q') \qquad p.3' : (3, ps') \to (2, p)(0, s')$$
$$p.4 : (0, qq') \to (3, q')(1, q) \qquad p.4' : (0, ss') \to (3, s')(1, s)$$
$$p.5 : (2, pZ) \to (0, p)(3, Z) \qquad p.6 : (3, ZZ') \to (2, Z)(0, Z')$$
$$p.7 : (0, Z'Z'') \to (3, Z')(1, Z'')$$

For any instruction (p, A_i-, q, s) of M we add the following rules to R_1:

$$p.1 : (1, pp_1) \to (3, p)(2, p_1) \qquad p.2 : (2, p_1A_i) \to (3, p_1)(0, A_i)$$
$$p.3 : (3, pp_2) \to (0, p)(2, p_2) \qquad p.4 : (3, p_1p_3) \to (1, p_1)(2, p_3)$$
$$p.5 : (2, p_1p_2) \to (1, p_1)(0, p_2) \qquad p.6 : (0, p_2s) \to (3, p_2)(1, s)$$
$$p.7 : (2, p_2p_3) \to (0, p_3)(3, p_2) \qquad p.8 : (0, p_3q) \to (3, p_3)(1, q)$$

Now we prove that any instruction of M can be simulated by a set of rules of Π. We first note that during the functioning of Π there is no more than one symbol $p \in Q$ in cell 1, and the simulation of any instruction of M can only start if an element of Q is in cell 1.

We start with instructions of the form (p, A_i+, q, s). The simulations starts with p in cell 1. Then rule $p.1$ is applied, after then $p.2$ and $p.3$ or p_2 and $p.3'$ are performed in parallel. At the end of this phase of the computation, p and one more copy of A_i will be in cell 2 and q' or s', respectively, moves to the environment. At the next moment, either rules $p.4$ and $p.5$ or $p.4'$ and $p.5$ are applied in parallel, resulting in symbols q or s in cell 1, p being sent out to the environment, and Z is in cell 3. Now, the simulation of a new instruction of M can start. However, two more rules are still applied in the next two steps, $p.6$ and $p.7$. These two rules provide Z in cell 2 and Z' in cell 3; these symbols will be needed later. Notice that the application of these rules does not interfere with the simulation of any instruction of M, thus they can be performed. We also note that during the computation symbols Z'' are accumulated in cell 1, but this fact does not influence the simulation. It is easy to see that the rules can be performed only in the order above and that this computation phase of Π corresponds to the execution of instruction (p, A_i+, q, s).

We continue with the simulation of instructions of the form (p, A_i-, q, s). At the first step, rule $p.1$ is applied which moves symbol p to cell 3 and symbol p_1 to cell 2. If cell 2 contains at least one copy of A_i, then in the next step rules $p.2$ and $p.3$ can be applied in parallel, otherwise only $p.3$ is applicable. Suppose that cell 2 contains at least one A_i. Then one copy of A_i and p leave the system, p_1 moves to cell 3 and p_2 to cell 2. After then rules $p.4$, $p.7$, and $p.8$ are applied, thus p_1 and symbol q enter cell 1, and symbols p_2 and p_3 return to their original location, namely to cell 3. If cell 2 does not contain any copy of A_i, then after

applying rule $p.3$, rules $p.5$ and $p.6$ are applied one after each other. Thus, the zero check has been simulated since cell 1 contains symbols s and p_1 and cell 3 has p_2 and p_3. The reader may easily see that these rules of Π can be applied only in the order given above. This means that they simulate the instruction (p, A_i-, q, s) of Π and only that.

By the previous considerations, we obtain that any computation in M can correctly be simulated by Π and conversely and $N(M) = N(\Pi)$ holds. Since $N(M)$ is a recursively enumerable set of numbers, the statement of the theorem is valid.

As in the previous cases, generalized communicating P systems with three cells and with only chain rules are computationally complete computing devices. In [3] computational completeness was proved, however no size bound was presented.

Theorem 3. $NOtP_3(chain) = NRE$.

Proof. Let us consider an arbitrary register machine $M = (Q, R, q_0, q_f, P)$ with $R = \{A_1, \dots A_n\}$, $n \geq 1$, given as in Section 2. To prove the statement, we construct a generalized communicating P system $\Pi = (O, E, w_1, w_2, w_3, R_1, 2)$ with chain rules such that $N(M) = N(\Pi)$. The proof is based on the bisimulation of M by Π, i.e., by showing that for any successful generation in M there exists a successful generation in Π and conversely such that the two generation processes yield the same number as result.

Let Q^+ and Q^- be the sets of labels of the increment instructions and the decrement instructions of M, respectively.

Let us define the alphabet of objects of Π as
$O = Q \cup R \cup \{p', \bar{p} \mid p \in Q\} \cup \{p_1, p_2 \mid p \in Q^-\} \cup \{Z\}$.

Let $E = Q \cup R \cup \{p' \mid p \in Q^+\} \cup \{p_1, p_2 \mid p \in Q^-\} \cup \{Z\}$. and $w_1 = \{q_0\} \cup \{\bar{p} \mid p \in Q\}$, $w_2 = \emptyset$, $w_3 = \emptyset$.

The set of rules R_1 of Π is defined as follows.

For any rule (p, A_i+, q, s) of M we add the following rules to R_1:

$$p.1 : (1,p)(0,p') \to (2,p)(1,p') \qquad p.2 : (1,p')(0,q) \to (2,p')(1,q)$$
$$p.2' : (1,p')(0,s) \to (2,p')(1,s) \qquad p.3 : (2,p)(0,A_i) \to (3,p)(2,A_i)$$
$$p.4 : (3,p)(2,p') \to (0,p)(3,p') \qquad p.5 : (0,Z)(3,p') \to (1,Z)(0,p')$$

For any rule (p, A_i-, q, s) of M we add the following rules to R_1:

$$p.1 : (1,p)(0,p_1) \to (3,p)(1,p_1) \qquad p.2 : (1,p_1)(0,p_2) \to (2,p_1)(1,p_2)$$
$$p.3 : (3,p)(2,A_i) \to (0,p)(3,A_i) \qquad p.4 : (3,A_i)(2,p_1) \to (0,A_i)(3,p_1)$$
$$p.5 : (3,p)(1,p_2) \to (0,p)(3,p_2) \qquad p.6 : (0,q)(3,p_1) \to (2,q)(0,p_1)$$
$$p.7 : (3,p_2)(1,\bar{s}) \to (0,p)(3,\bar{s}) \qquad p.8 : (1,p_2)(2,q) \to (0,p_2)(1,q)$$
$$p.9 : (0,s)(3,\bar{s}) \to (1,s)(0,\bar{s}) \qquad p.10 : (0,\bar{s})(2,p_1) \to (1,\bar{s})(0,p_1)$$

We show that any instruction of M can be simulated by applying rules of Π, and conversely, any successful computation in Π corresponds to a successful computation in M. We note that as in the proofs of the previous theorems, cell 1 contains at most one symbol that corresponds to a state of M.

Let us consider instructions of M of the form (p, A_i+, q, s). To simulate this instruction, first rule $p.1$ and then either rule $p.2$ or rule $p.2'$ is applied (depending on whether the next state of M will be q or s) in parallel with rule $p.3$. Then, the number of symbols A_i in cell 2 is increased by one and the symbol representing the new state, i.e., q or s enters cell 1. (During the work of Π, cell 2 will store symbols A_i which represent the contents of the corresponding register.) Still we need to remove p and p' from the system. This is done by rules $p.4$ and $p.5$. Notice that the simulation of a new instruction may start before the application of rules $p.4$ and $p.5$, but these two rules do not interfere with any such computation phase, so these two rules may be applied. It can also be seen that the above rules, performed in the above order, simulate instruction (p, A_i+, q, s) and only that. We note that symbols Z are accumulated in cell 1, but the presence of these symbols in cell 1 has no effect on the simulation.

Now let us consider instructions of the form (p, A_i-, q, s). First rule $p.1$ is applied and then either rules $p.2$ and $p.3$ are applied in parallel (cell 2 contains at least one A_i) or only rule $p.2$ can be applied (cell 2 does not contain A_i). Suppose that cell 2 has at least one symbol A_i. Then, by applying rules $p.4$, $p.5$, $p.6$ and $p.8$ in this order, one copy of A_i leaves the system, symbol q enters the system and finally moves to cell 1, and assistant symbols p_1 and p_2 leave the systems as well. No other rule can be applied during this phase of the computation. Suppose now that cell 2 does not contain any occurrence of A_i. Then, only rule $p.2$ can be applied. Then, p is present in cell 3, p_1 in cell 2, and p_2 in cell 1. After then, rules $p.5$, $p.7$, $p.9$ and $p.10$ are applied in this order. As a result, s enters the system and enters cell 1, assistant symbols p_1 and p_2 leave the system, and the further assistant symbol $\bar{s}$ enters cell 1 and remains there. The presence of this symbol in cell 1 and the fact that rule $p.10$ is applied after s reaches its destination, cell 1, do not affect the computation.

We may easily notice that the rules can be performed only in the manner described above. Thus, any computation in M can correctly be simulated by Π and conversely, and $N(M) = N(\Pi)$ holds. Since $N(M)$ is a recursively enumerable set of numbers, the theorem holds.

4 Conclusions

In this paper we proved that GCPSs with three cells and with only join, or only split, or only chain rules are computationally complete computing devices. These bounds are improvements of the previous results. We guess that the number of cells can also be significantly reduced (to 3) in the case of other variants of generalized communicating P systems with minimal interaction, we plan investigations in this direction in the near future.

5 Acknowledgment

The work of E. CS-V. was supported by NKFIH (National Research, Development, and Innovation Office), Hungary, Grant no. K 120558.

References

1. Á. Balaskó, E. Csuhaj-Varjú, Gy. Vaszil, Dynamically Changing Environment for Generalized Communicating P Systems. In: G. Rozenberg et. al (Eds.) Membrane Computing - 16th International Conference, CMC 2015, Valencia, Spain, August 17-21, 2015, Revised Selected Papers. LNCS 9504, Springer, 2015, 92-105.
2. E. Csuhaj-Varjú, Gy. Vaszil and S. Verlan, On generalized communicating P systems with one symbol. In M. Gheorghe et al. (Eds.) 11th International Conference on Membrane Computing, 2010, Jena, Germany, LNCS 6501, Springer, 2010, 160-174.
3. E. Csuhaj-Varjú and S. Verlan, On generalized communicating P systems with minimal interaction rules. Theoretical Computer Science, 412, 2011, 124-135.
4. R. Freund and S. Verlan, A formal framework for static (tissue) P systems. In G. Eleftherakis et. al (Eds.) Membrane Computing, 8th International Workshop, WMC 2007, Thessaloniki, Greece, June 25-28, 2007 Revised Selected and Invited Papers, LNCS 4860, Springer, 2007, 271-284.
5. S.N. Krishna, M. Gheorghe and C. Dragomir, Some classes of generalised communicating P systems and simple kernel P systems. Technical report, CS-12-03, University of Sheffield, 2013; available at http://staffwww.dcs.shef.ac.uk/people/M.Gheorghe/research/paperlist.html
6. S.N. Krishna, M. Gheorghe and C. Dragomir, Some classes of generalised communicating P systems and simple kernel P systems. In P. Bonizzoniet al. (Eds) The Nature of Computation. Logic, Algorithms, Applications - 9th Conference on Computability in Europe, CiE 2013, Milan, Italy, 2013, LNCS 7921, Springer, 2013, 284-293.
7. S. N. Krishna, M. Gheorghe, F. Ipate, E. Csuhaj-Varjú, R. Ceterchi, Further results on generalised communicating P systems. Theoretical Computer Science, in press. Available online 1 June 2017, https://doi.org/10.1016/j.tcs.2017.05.020
8. M. Minsky, Finite and Infinite Machines. Prentice Hall, Englewood Cliffs, New Jersey, 1967
9. G. Păun, G. Rozenberg, A. Salomaa (Eds.) The Oxford Handbook of Membrane Computing. Oxford University Press, 2010
10. G. Rozenberg and A. Salomaa (Eds.) Handbook of Formal Languages, volume 1-3, Springer, 1997
11. S. Verlan, F. Bernardini, M. Gheorghe, M. Margenstern, On communication in tissue P systems: conditional uniport. In: Membrane Computing, 7th International Workshop, WMC 2006, Leiden, The Netherlands, July 17-21, 2006, Revised, Selected, and Invited Papers. LNCS 4361, Springer, 2006, 521-535.
12. S. Verlan, F. Bernardini, M. Gheorghe, M. Margenstern, Generalized communicating P systems. Theoretical Computer Science 404(1-2), 2008, 170-184.