Journal of Universal Computer Science, vol. 10, no. 5 (2004), 509-539
submitted: 1/4/04, accepted: 10/4/04, appeared: 28/5/04 © J.UCS

Population P Systems

Francesco Bernardini, Marian Gheorghe
(University of Sheffield, UK
{F.Bernardini, M.Gheorghe}@dcs.shef.ac.uk)

Abstract: This paper introduces a notion of population P systems as a class of tissue P systems where the links between the cells can be modified by means of a specific set of bond making rules. As well as this, cell division rules which introduce new cells into the system, cell differentiation rules which change the set of rules that can be used inside of a cell, and cell death rules which remove cells from the system are also considered by introducing a particular notion of population P systems with active cells. The paper mainly reports universality results for the following models: (a) population P systems where cells are restricted to communicate only by means of the environment but never forming any bond; (b) population P systems with bond making rules with restricted communication rules; (c) population P systems possessing only the cell differentiation operation; and (d) population P systems equipped with cell division rules and bond making rules.

Key Words: Membrane computing, Cell bonding, Cell division, Cell differentiation, Turing computability

Category: F.1.1, F.4.3

1 Introduction

Membrane computing represents a new and rapidly growing research area which is part of the natural computing paradigm. Already a monograph has been dedicated to this subject [Păun 2002] and some fairly recent results can be found in [Păun et al. 2003], [Martín-Vide et al. 2004]. Membrane computing has been introduced with the aim of defining a computing device, called P system, which abstracts from the structure and the functioning of living cells [Păun 2000]. Membranes are among the main elements of the living cells; they separate the cell from its environment and split the content of the cell into small compartments by means of internal membranes. Each compartment contains its own enzymes and their specialized molecules. Therefore, a membrane structure has been identified as the main characteristic of every P system that is defined as a hierarchical arrangement of different membranes embedded in a unique main membrane that identify several distinct regions inside the system. Each region gets assigned a finite multiset of objects and a finite set of rules either modifying the objects or moving them from a place to another one. The structure of a P system is usually represented as a tree describing the hierarchical architecture based on membranes.

A natural generalization of the P system model can be obtained by considering P systems where the structure of the system is defined as an arbitrary graph. Each node in the graph represents a membrane, which gets assigned a multiset of objects and a set of rules for modifying these objects and communicating them alongside the edges of the graph [Păun 2002]. These networks of communicating membranes are also known as tissue P systems because, from a biological point view, they can be interpreted as an abstract model of multicellular organisms. In such organisms, cells are specialized members of a multicellular community. They collaborate with each other to form a multitude of different tissues, arranged into organs performing various functions [Alberts et al. 2002]. This model may be also regarded as an abstraction of a population of bio-entities aggregated together in a more complex bio-unit. In this respect the model addresses not only the cellular and tissue levels, but also the case of various colonies of more complex organisms like ants, bees etc. We have to use in this paper the notion of population instead of tissue in order to cover both modelling aspects mentioned above, although the tissue context will predominantly be used throughout the paper. These populations of individuals are usually far from being stable; mechanisms enabling new individuals to be introduced, to update the links between them or to remove some individuals play a fundamental role in the evolution of a biological system as a population of interacting/cooperating components. By starting from these observations, a framework to develop population P systems has been recently proposed in [Bernardini and Gheorghe 2004a] as a class of tissue P systems relying on cell division as a mechanism to modify the structure of the underlying graph.

In this paper we present a variant of population P systems by considering a model where the structure of the underlying graph is continuously modified after each step of transformation-communication according to a specific set of bond making rules, which are able to add/remove edges from the graph defining the current structure of the system. A transformation-communication step mainly consists in applying rules for modifying the objects and communicating them from a cell to another one. In particular, communication rules are of the forms considered in [Bernardini and Gheorghe 2004b], which are inspired by the general mechanism of cell communication based on signals and receptors. As well as this, cell division rules for duplicating the existing nodes in the graph, cell differentiation rules for changing the type of the cells associated with the nodes in the graph, and cell death rules for removing nodes from the graph can also be applied in a transformation-communication step. Finally, another important feature of the model considered here is the notion of environment as a repository of objects that are sent out from the cells. The objects in the environment can subsequently re-enter the cells, and this provides a form of undirect communication among the system's cells.

The power of population P systems is here investigated by considering separately the case when the stucture of the underlying graph cannot be modified by any rule, the case when only the edges of the graph can be modified by using a finite set of bond making rules, and the case when both the edges and the nodes in the graph can be modified by allowing the cells to use cell division rules, cell differentiation rules, and cell death rules. In the former case, the main result obtained here shows the universality for completely unstructured P systems where cells can communicate only indirectly by means of the environment without never forming any bond. Next, when bond making rules are considered, population P systems are proved to be computationally complete even when simple communication rules where objects can be moved from a cell to another one without any restriction are considered. Finally, we provide an universality result for population P systems that use only the operation of cell differentiation and an universality results for population P systems that use cell division in combination with bond making rules.

2 Preliminaries

We recall here some basic notions and notation commonly used in membrane computing and the few notions of formal language theory we need in the rest of the paper. We refer to [Păun 2002], [Rozenberg and Salomaa 1997] for further details.

An alphabet is a finite non-empty set of abstract symbols. Given an alphabet V, we denote by V^* the set of all possible strings over V, including the empty string λ. The length of a string $x \in V^*$ is denoted by $|x|$ and, for each $a \in V$, $|x|_a$ denotes the number of occurrences of the symbol a in x. A multiset over V is a mapping $M : V \longrightarrow N$ such that, $M(a)$ defines the multiplicity of a in the multiset M (N denotes the set of natural numbers). Such a multiset can be represented by a string $a_1^{M(a_1)}\,a_2^{M(a_2)}\ldots a_n^{M(a_n)} \in V^*$ and by all its permutations with $a_j \in V$, $M(a_j) \neq 0$, $1 \leq j \leq n$. In other words, we can say that each string $x \in V^*$ identifies a finite multiset over V defined by $M_x = \{\,(a, |x|_a) \,|\, a \in V\,\}$. Moreover, given two strings $x, y \in V^*$, we denote by xy their catenation, which corresponds to the union of the multiset represented by string x and the multiset represented by string y.

In the following proofs we will use the notion of counter machines in the form considered in [Bernardini and Păun 2004], [Frisco and Hoogeboom 2003]. Informally, a counter machine is a finite state machine that has a finite number of counters able to store values represented by natural numbers; the machine runs a program consisting of instructions which can increase or decrease by one the contents of registers, changing at the same time the state of the machine; starting with each counter empty, the machine performs a computation; if it reaches a terminal state, then the number stored in a specified counter is said to be generated during this computation. It is known that counter machines (of various types) are computationally universal in the sense that they can generate the whole family of Turing computable sets of natural numbers [Hopcroft and Ulmann 1979].

Definition 1. A counter machine is a construct

$$M = (Q, F, p_0, A, c_{out}, I),$$

where:

1. Q is the set of states,
2. $F \subseteq Q$ is the set of final states,
3. $p_0 \in Q$ is the initial state,
4. A is the set of counters,
5. $c_{out} \in A$ is the output counter,
6. I is a finite set of instructions of the following forms:
   (a) $(p \to q, +c)$, with $p, q \in Q,\ c \in A$: add 1 to the value of the counter c and move from state p into state q;

   (b) $(p \to q, -c)$, with $p, q \in Q,\ c \in A$: if the current value of the counter c is not zero, then subtract 1 from the value of the counter c and move from state p into state q; otherwise the computation is blocked in state p;

   (c) $(p \to q, c = 0)$, with $p, q \in Q,\ c \in A$: if the current value of the counter c is zero, then move from state p into state q; otherwise the computation is blocked in state p.

   (d) $(p \to q, \epsilon)$, with $p, q \in Q$: move from state p into state q without changing the value of any counter.

A transition step in such a counter machine consists in updating/checking the value of a counter according to an instruction of one of the types presented above and moving from a state to another one. Starting with the number zero stored in each counter, we say that the counter machine computes the value n if and only if, starting from the initial state, the system reaches a final state after a finite sequence of transitions, with n being the value of the output counter c_{out} at that moment. Without loss of generality, we may assume that in the end of the computation the machine makes zero all the counters but the output counter as well as that there are no transitions that start from a final state. As we have mentioned above, such counter machines when equipped with at least three counters are computationally equivalent to Turing machines, and we will make below an essential use of this result. In this respect, we denote by NRE the family of recursively enumerable sets of natural numbers that can be computed by Turing machines or counter machines.

As well as this, we need the notion of extended tabled 0L system (ET0L system), which is a construct $G = (V, T, w, P_1, \ldots, P_m)$, $m \geq 1$, where V is an alphabet, $T \subseteq V$, $w \in V^*$, and P_i, $1 \leq i \leq m$, are finite sets of rules (tables) of context-free rules over V of the form $a \to x$. In a derivation step, all the symbols present in the current sentential form are rewritten using one table. The language generated by G, denoted by $L(G)$, consists of all the strings over T which can be generated in this way by starting from w. An ET0L system with only one table is called an E0L system. We denote by $E0L$ and $ET0L$ the families of languages generated by E0L systems and ET0L systems, respectively. Furthermore, we denote by $NE0L$ and $NET0L$ the families of length sets associated with languages in $ET0L$ and $E0L$, respectively. It is known from [Rozenberg and Salomaa 1980] that $NCF \subset NE0L \subset NET0L \subset NCS$, with NCF the family of length sets of context-free languages, and NCS the family of length sets of context-sensitive languages. Moreover, according to Theorem 1.3 in [Rozenberg and Salomaa 1980], for each language $L \in ET0L$ there is an ET0L system that generates L, that contains only two tables, that is, $G = (V, T, w, P_1, P_2)$, and where, after having used P_1, we can use any of P_1 and P_2 but, after having used P_2, we always use P_1.

Finally, we recall here the notion of P systems with catalysts from [Păun 2002].

Definition 2. A P system with catalysts is a construct

$$\Pi = (V, C, \mu, w_1, w_2, \ldots, w_n, R_1, R_2, \ldots, R_m, i_o),$$

where:

1. V is a finite set of symbols called objects;
2. $C \subseteq V$ is a finite set of objects called catalysts;
3. μ is a membrane structure consisting of m membranes, with the membranes (and hence the regions) injectively labeled by $1, 2, \ldots, m$;
4. for each $1 \leq i \leq n$, R_i is a finite set of evolution rules that is associated with region i in μ; an evolution rule is either of the form $a \to v$, or of the form $c\,a \to c\,v$, with $c \in C$, $a \in (V - C)$, $v \in ((V - C) \times \{here, out, in\})^*$;
5. $i_o \in \{1, 2, .., m\}$ is the label of an elementary membrane that identifies the output membrane.

The basic feature of a P system with catalysts is the membrane structure μ that has to be considered as a hierarchical arrangement of m distinct membranes embedded in a unique main membrane called the skin membrane. This membrane structure is usually represented as a string of pairs of matching square brackets, which are labeled in an one-to-one manner by $1, 2, \ldots, m$. Each pair of square brackets represents a membrane (membrane i) with its corresponding region (the region delimited by membrane i, or region i). Moreover, this representation makes possible to point out the relationships of inclusions among membranes and regions: we say a region i contains a membrane j if and only if the pair of square brackets labeled by i embraces the pair of square brackets labeled by j.

Then, each region i gets assigned a finite multiset of objects w_i, which defines the initial content of the membrane i, and a finite set of evolution rules R_i. An evolution rule in R_i is either of the form $a \rightarrow v$, or of the form $ca \rightarrow cv$ (*catalytic rules*). In the former case, the rule specifies that, inside region i, an object a can be consumed in order to produce a new multiset v. A catalytic rule instead specifies that, inside region i, an object a can be consumed in order to produce a new multiset v only in the presence of an occurrence of the catalyst c. A particular occurrence of a catalyst can be used only by one catalytic rule at a time, and catalysts are never modified by any rule in R_i. However, an evolution rule in R_i always produces a new multiset v that contains pairs of the form (b, t), with $b \in (V - C)$ an object that is not a catalyst, and $t \in \{here, in, out\}$ a target indication that specifies where the object b has to be moved. Specifically, *here* means the object b has to stay in the region where the rule is applied, *out* means that the object b has to exit from the region where the rule is applied, and *in* means the object b has to enter one of the membranes contained in the region where the rule is applied; that membrane is non-deterministically chosen.

As usual, by starting from the initial configuration, a computation is obtained by applying to the objects contained in the various regions the corresponding set of rules in a maximally parallel manner. A computation is said to be successful if it reaches a configuration where no more rules can be applied to the objects in the system. The result of successful computation is given by the natural number obtained by counting the objects that are present in the output region i_o in the final configuration. The set of natural numbers generated in this way by all the successful computations occurring in a P system Π is denoted by $N(\Pi)$. Moreover, we denote by $NOP_m(Cat_n)$, with $n, m \geq 0$, the family of sets of natural numbers generated by P systems with at most m membranes and that use at most n different catalysts.

The main result concerning the power of P systems with catalysts is the following one [Freund et al. 2003].

Theorem 3. $NOP_2(Cat_2) = NRE$.

This means P systems with at most 2 membranes that use at most 2 different catalysts are computationally complete in the sense that they are able to generate the whole family of sets of natural numbers generated by Turing machines or counter machines.

A particular class of P systems is represented by the class of P systems with no catalysts where all the rules are of the form $a \to v$. These systems are also called non-cooperative P systems and the families of sets of natural numbers generated by non-cooperative P systems are usually denoted by $NOP_m(nCoo)$, with $m \geq 0$. In this case, the main result from [Păun 2002] concerning the power of non-cooperative P systems is:

Theorem 4. $NOP_*(nCoo) = NOP_1(nCoo) = NCF.$

The proof of this result can be found in Therefore, in the case of non-cooperative P systems, the hierarchy on the number of membranes collapses at level one and systems of this form are able to generate only length sets of context-free languages.

3 Population P Systems

We introduce here a notion of population P systems as a finite collection of different cells that are able of forming/removing bonds according to a finite set of bond handling rules.

Definition 5. A population P system is a construct

$$\mathcal{P} = (V, \gamma, \alpha, w_e, C_1, C_2, \ldots, C_n, c_o),$$

where:

1. V is a finite alphabet of symbols called objects;
2. $\gamma = (\{1, 2, \ldots, n\}, E)$, with $E \subseteq \{\{i, j\} \mid 1 \leq i \neq j \leq n\}$, is a finite undirected graph;
3. α is a finite set of bond making rules $(i, x_1; x_2, j)$, with $x_1, x_2 \in V^*$, and $1 \leq i \neq j \leq n$;
4. $w_e \in V^*$ is a finite multiset of objects initially assigned to the environment;
5. $C_i = (w_i, S_i, R_i)$, for each $1 \leq i \leq n$, with:
   (a) $w_i \in V^*$ a finite multiset of objects,
   (b) S_i is a finite set of communication rules; each rule has one of the following forms: $(a; b, in)$, $(a; b, enter)$, $(b, exit)$, for $a \in V \cup \{\lambda\}$, $b \in V$,
   (c) R_i is a finite set of transformation rules of the form $a \to y$, for $a \in V$, and $y \in V^+$;
6. c_o is the (label of the) output cell, $1 \leq c_o \leq n$.

A population P system $\mathcal{P}$ is defined as a collection of n cells where each cell C_i corresponds in an one-to-one manner to a node i in a finite undirected graph γ, which defines the initial structure of the system. Cells are allowed to communicate alongside the edges of the graph γ, which are unordered pairs of the form $\{i, j\}$, with $1 \leq i \neq j \leq n$. The cells C_i, $1 \leq i \leq n$, are associated in a one-to-one manner with the set of nodes $\{1, 2, \ldots, n\}$. For this reason, each cell C_i will be subsequently identified by its label i from the aforementioned set.

Then, each cell C_i gets assigned a finite multiset of objects w_i, a finite set of communication rules S_i, and a finite set of transformation rules R_i.

Each set R_i contains rules of the form $x \to y$ that allow cell i to consume a multiset x in order to produce a new multiset y inside cell i.

Communication rules in S_i of the form $(a; b, in)$ are instead used by cell i to receive objects from its neighbouring cells. In fact, a rule $(a; b, in)$ in S_i means that, in the presence of an object a inside the cell i, an object b can be moved from a cell j non-deterministically chosen to the cell i, given that $\{i, j\} \in A$. In particular rules of the form $(\lambda; a, in)$ allow a cell to receive an object b from any of the cells linked to cell i at any moment without any restriction. Moreover, for each cell i, rules of the form $(a, exit)$ are also considered in S_i; these rules allow the cell i to release an object a in the environment. Objects from the environment can enter the cell i by means of rules of the forms $(a; b, enter)$ (i.e., an object b in the environment can enter cell i only in the presence of another object a), and $(\lambda; b, enter)$ (i.e., at any time an object b can enter cell i without any specific restriction). The objects that are currently associated with the environment are kept in a distinct multiset that is initially defined by w_e.

Cell capability of moving objects alongside the edges of the graph is then influenced by particular bond making rules in α that allow cells to form new bonds. In fact, a bond making rule $(i, x_1; x_2, j)$ specifies that, in the presence of a multiset x_1 in the cell i and a multiset x_2 inside the cell j, a new bond can be created between these two cells. This means a new edge $\{i, j\}$ can be added to the graph that currently defines the structure of the system.

Therefore, a population P system is basically defined as a tissue P system in the form introduced in [Bernardini and Gheorghe 2004b] with the important difference that now the underlying structure of the system can be changed by using a specific set of bond making rules. As well as this, cells in a population P system are also allowed to communicate indirectly by means of the environment.

As the reader can easily notice, there are several similarities and differences between our systems and the so-called evolution-communication P systems introduced in [Cavaliere 2002]: evolution-communication P systems also use separate rules for multiset rewriting and object communication, but the latter rules are usual symport/antiport rules, the framework is that of cell-like P systems, and the membrane structure is rigid.

A step of a computation in a population P system $\mathcal{P}$ is defined as being performed in two separate stages: the content of the cells is firstly modified by applying the communication rules in S_i, and the transformation rules in R_i, for all $1 \leq i \leq n$; the structure of the system is then modified by using the bond making rules in α. Specifically, a configuration of a population P system $\mathcal{P}$ at any time is given by a tuple Σ such that:

$$\Sigma = \langle \gamma', w'_e, w'_1, w'_2, \ldots, w'_n \rangle,$$

with $\gamma' = (\{1, 2, \ldots, n\}, E')$, for $E' \subseteq \{\{i, j\} \mid 1 \leq i \neq j \leq n\}$ (the graph that defines the current structure of the system $\mathcal{P}$), $w'_e \in V^*$ (the multiset of objects that are currently associated with the environment) and, for all $1 \leq i \leq n$, $w'_i \in V^*$ (the multiset of objects that defines the current content of cell i). Thus, given three configurations Σ', Σ'', Σ''' such that:

$$\begin{aligned} \Sigma' &= \langle \gamma', w'_e, w'_1, w'_2, \ldots, w'_n \rangle, \\ \Sigma'' &= \langle \gamma'', w''_e, w''_1, w''_2, \ldots, w''_n \rangle, \\ \Sigma''' &= \langle \gamma''', w'''_e, w'''_1, w'''_2, \ldots, w'''_n \rangle, \end{aligned}$$

we write $\Sigma' \overset{tc}{\Longrightarrow}_{\mathcal{P}} \Sigma''$ and $\Sigma'' \overset{bm}{\Longrightarrow}_{\mathcal{P}} \Sigma'''$, if and only if the following conditions hold.

- $\gamma' = \gamma''$, and $w''_e, w''_1, w''_2, \ldots, w''_n$ are multisets of objects obtained by applying, in a non-deterministic maximal parallel manner, the communication rules in $S_1, S_2, \ldots, S_n$, and the transformation rules in $R_1, R_2, \ldots, R_n$ to the multisets $w'_e, w'_1, w'_2, \ldots, w'_n$; $\overset{tc}{\Longrightarrow}_{\mathcal{P}}$ is meant to be a transformation-communication relationship.
- $w'''_e = w''_e$, $w'''_i = w''_i$, for all $1 \leq i \leq n$, and γ''' is obtained by removing all the edges from γ'' and adding an edge $\{i, j\}$, for each $1 \leq i \neq j \leq n$ such that there exists a bond making rule $(i, x_1; x_2, j) \in \alpha$, with x_1 a multiset that is contained in w''_i, and x_2 a multiset that is contained in w''_j; $\overset{bm}{\Longrightarrow}_{\mathcal{P}}$ is meant to be a bond making relationship.

In this way, by combining the relations $\overset{tc}{\Longrightarrow}_{\mathcal{P}}$, $\overset{bm}{\Longrightarrow}_{\mathcal{P}}$, we say the population P systems $\mathcal{P}$ is able to transit from a configuration Σ' to a configuration Σ''' in a single step of a computation, and we write $\Sigma' \Longrightarrow_{\mathcal{P}} \Sigma'''$.

A successful computation in $\mathcal{P}$ is then defined as a finite sequence of transitions of the form

$$\Sigma_0 \Longrightarrow_{\mathcal{P}} \Sigma_1 \Longrightarrow_{\mathcal{P}} \ldots \Longrightarrow_{\mathcal{P}} \Sigma_{k-1} \Longrightarrow_{\mathcal{P}} \Sigma_k,$$

where $k \geq 1$, Σ_0 is the initial configuration of the system $\mathcal{P}$ as specified in Definition 5, and Σ_k is a final configuration such that there does not exist any configuration $\Sigma' \neq \Sigma_k$, with $\Sigma_k \stackrel{tc}{\Longrightarrow}_{\mathcal{P}} \Sigma'$. In other words, a successful computation is a computation that halts in a configuration where, after a last bond making stage, the content of the cells cannot be modified anymore by means of some communication or transformation rules. The result of a successful computation is given by the number of objects that are placed inside the output cell c_o in the final configuration. The set of natural numbers that are generated in this way by all the successful computations in $\mathcal{P}$ is denoted by $N(\Pi)$.

The generative capacity of population P systems is investigated in the next two subsections by considering separately the case when the feature of bond making is not considered and the case when it is. To this aim, we introduce the families of sets of natural numbers $NOPP_{n,k}(c, b)$, with $n \geq k \geq 1$, $c \in \{nR, R\}$, and $b \in \{n\alpha, \alpha_t \mid t \geq 0\}$, which are generated by population P systems where:

- the number of cells in the system is less than or equal to n;
- in each step of a computation, the number of cells in each connected component of the graph defining the structure of the system is always less than or equal to k; a connected component of a graph is any subset of nodes such that, given any two nodes in that subset, there always exists a path that connects them, and there does not exists any path from these nodes to nodes that are not in that subset;
- $c = nR$ specifies that all the communication rules that are associated with the cells in the system are of the form $(\lambda; b, in)$, $(\lambda; b, enter)$, $(b, exit)$;
- $c = R$ specifies that communication rules of any form are allowed to be used inside the cells;
- $b = n\alpha$ specifies that bond making rules are not considered and the structure of system is given by the initial graph, which is never modified during a computation;
- $b = \alpha_t$, for some $t \geq 0$, specifies that all the bond making rules in the system are of the form $(i, x_1; x_2, j)$, with $|x_1| \leq t$ and $|x_2| \leq t$; as well as this, we say that a bond making rule $(i, x_1; x_2, j)$ is of size $h = \max\{|x_1|, |x_2|\}$.

Finally, if the value of n, k, or t is not specified, then it is replaced by the symbol $*$.

3.1 The Power of Evolution-Communication

Here we present some results concerning the power of population P systems as evolution-communication P systems where the underlying structure is never modified during a computation.

In the case of simple communication rules of the forms $(\lambda; b, in)$, $(\lambda; b, enter)$, $(b, exit)$, the hierarchy on the number of membranes collapses at level one. Moreover, the corresponding family of sets of natural numbers coincides with the family of length sets of context-free languages (i.e., the family of semilinear sets of numbers).

Lemma 6. $NOPP_{*,*}(nR, n\alpha) = NOPP_{1,1}(nR, n\alpha) = NCF$.

Proof. The proof can be derived in a straightforward manner from the proof of Theorem 4 that concerns the power of non-cooperative P systems. In fact, let $\mathcal{P}$ be a population P system such that:

$$\mathcal{P} = (V, \gamma, w_e, (w_1, S_1, R_1), (w_2, S_2, R_2), \ldots, (w_n, S_n, R_n), c_o),$$

with $n \geq 1$, and all the symbols as in Definition 5. For each $1 \leq i \leq n$, we define the set T_i that contains all the symbols $a \in V$ such that:

- there does not exist any rule $a \to y \in R_i$, and there does not exist any rule $(\lambda; a, exit) \in S_i$;
- for all edges in γ of the form $\{i, j\}$, with $1 \leq i \neq j \leq n$, there does not exist any rule $(\lambda; a, in) \in S_j$.

Thus, for each $1 \leq i \leq n$, the set T_i contains all the symbols that cannot be used in any rule once introduced in cell i. In a similar way, we define the set T_e that contains all the symbols $a \in V$ such that, for all $1 \leq i \leq n$, there does not exist any rule $(\lambda; a, enter) \in R_i$.

Next, let $\dagger$ be a new symbol that is not in V. For each $1 \leq i \leq n$, we define the homomorphism h_i such that: $h_i(a) = a_i$, if $a \in (V - T_i)$, $h_i(a) = \dagger$, if $i \neq c_o$ and $a \in T_i$, $h_i(a) = a$, if $i = c_o$ and $a \in T_{c_o}$. As well as this, we define the homomorphism h_e such that: $h_e(a) = a_e$, if $a \in (V - T_e)$, $h_e(a) = \dagger$, if $a \in T_e$.

Now, we can construct a population P systems $\mathcal{P}'$ that is able to simulate the behaviour of $\mathcal{P}$ such that

$$\mathcal{P}' = (V', \gamma, \lambda, (w'_1, S'_1, R'_1), c_o),$$

where:

$$\begin{aligned}
V &= \{\, a_i \mid a \in (V - (T_e \cup T_1 \cup T_2 \cup \ldots \cup T_n)), 1 \leq i \leq n \,\} \cup T_{c_o} \cup \{\dagger\},\\
\gamma &= (\{1\}, \emptyset),\\
w'_1 &= h_e(w_e)h_1(w_1)h_2(w_2)\ldots h_n(w_n),\\
S'_1 &= \{(\dagger, exit)\},\\
R'_1 &= \{\, a_i \to h_i(y) \mid a \to y \in R_i, 1 \leq i \leq n \,\}\\
&\quad \cup \{\, a_i \to h_e(a) \mid (a, exit) \in S_i, 1 \leq i \leq n \,\}
\end{aligned}$$

$$\begin{aligned}
&\cup \{ a_e \to h_i(a) \mid (\lambda; a, in) \in S_i, 1 \leq i \leq n \} \\
&\cup \{a_i \to h_j(a) \mid (\lambda; a, in) \in S_j, 1 \leq i \neq j \leq n, \\
&\quad \text{there exists an edge } \{i, j\} \text{ in } \gamma\}, \\
c_o &= 1.
\end{aligned}$$

Now, with this construction, it is easy to see that $N(\mathcal{P}') = N(\mathcal{P})$. Moreover, it appears clearly that the population P systems $\mathcal{P}'$ is equivalent to a basic P system with one membrane that uses non-cooperative rules. On the other hand, it is obvious that any basic P system with one membrane that uses non-cooperative rules can be simulated by a population P system with one cell. This means the same result stated in Theorem 4 holds for population P systems with one cell that use restricted communication rules. □

The next result instead shows that, when communication rules of any form are allowed to be used inside the cells, population P systems are computationally complete and the hierarchy on the number of cells collapses at level two.

Theorem 7. $NOPP_{2,2}(R, n\alpha) = NRE$.

The proof of this result can be found in [Bernardini and Gheorghe 2004b]. However, in the context of population P systems where there are no bonds among the cells, which are therefore limited to communicate by means of the environment only, the problem of establishing the power of these systems is not obvious. This means considering systems where the number of cells in a connected component of the graph is always one. In order to answer this problem, we show that these particular unstructured population P systems are enough powerful to simulate P systems with catalysts.

Theorem 8. $NOP_*(Cat_n) \subseteq NOPP_{n+2,1}(R, n\alpha)$, *for all* $n \geq 0$.

Proof. This result states that the number of cells that are necessary to simulate a P system with catalysts does not depend on the number of membranes but it depends only on the number of different catalysts that are considered. Therefore, for the sake of simplicity, we just consider the case of P systems with two membranes that use two different catalysts. It is then left to the reader the task of generalising the present construction to the case of an arbitrary number of membranes and an arbitrary number of catalysts.

Let Π be a P system with catalysts as specified in Definition 2 such that

$$\Pi = (V, \{c', c''\}, [_1\, [_2\,]_2\,]_1, w_1, w_2, R_1, R_2, 2),$$

with $\{c', c''\} \subseteq V$. For each $1 \leq i \leq 2$, we define the set T_i that contains all the symbols $a \in (V - \{c', c''\})$ such that there does not exist any rule $a \to v \in R_i$, or any rule $ca \to cv \in R_i$, with $c \in \{c', c''\}$. This means, for each $1 \leq i \leq 2$, the set T_i contains the objects that cannot evolve anymore once introduced into the region delimited by membrane i.

Next, we consider the set of symbols

$$N = \{ a_i, a'_i \mid a \in (V - (\{c', c''\} \cup T_1 \cup T_2)), i \in \{1, 2\}\} \cup \{\dagger\},$$

and we introduce the following homomorphisms:

- $h_1 : V \longrightarrow N \cup \{\lambda\}$, with $h_1(a) = a_1$, if $a \in (V - (\{c', c''\} \cup T_1))$, $h_1(a) = \lambda$, if $a \in T_1$, and $h_1(c) = \lambda$, if $c \in \{c', c''\}$;
- $h_2 : V \longrightarrow N \cup T_2 \cup \{\lambda\}$, with $h_2(a) = a_2$, if $a \in (V - (\{c', c''\} \cup T_2))$, $h_2(a) = a$, if $a \in T_2$, and $h_2(c) = \lambda$, if $c \in \{c', c''\}$;
- $g'_1 : V \longrightarrow \{c'_1\} \cup \{\lambda\}$, with $g'_1(a) = \lambda$, if $a \neq c'$, and $g'_1(c') = c'_1$;
- $g'_2 : V \longrightarrow \{c'_2\} \cup \{\lambda\}$, with $g'_2(a) = \lambda$, if $a \neq c'$, and $g'_2(c') = c'_2$;
- $g''_1 : V \longrightarrow \{c'_1\} \cup \{\lambda\}$, with $g''_1(a) = \lambda$, if $a \neq c''$, and $g'_1(c') = c'_1$;
- $g''_2 : V \longrightarrow \{c'_2\} \cup \{\lambda\}$, with $g''_2(a) = \lambda$, if $a \neq c''$, and $g'_2(c'') = c''_2$;
- $t_1 : ((V - \{c', c''\}) \times \{here, in, out\}) \rightarrow N$, with
  - $t_1((a, here)) = \dagger$, if $a \in T_1$,
  - $t_1((a, here)) = a'_1$, if $a \in (V - (\{c', c''\} \cup T_1))$,
  - $t_1((a, out)) = \dagger$,
  - $t_1((a, in)) = a$, if $a \in T_2$,
  - $t_1((a, in)) = a'_2$, if $a \in (V - (\{c', c''\} \cup T_2))$;
- $t_2 : ((V - \{c', c''\}) \times \{here, out\}) \rightarrow N \cup T_2$, with
  - $t_2((a, here)) = a$, if $a \in T_2$,
  - $t_2((a, here)) = a'_2$, if $a \in (V - (\{c', c''\} \cup T_2))$,
  - $t_2((a, out)) = \dagger$, if $a \in T_1$,
  - $t_2((a, out)) = a'_1$, if $a \in (V - (\{c', c''\} \cup T_1))$.

At this point, we can construct a population P system $\mathcal{P}$ that is able to simulate the P system Π such that

$$\mathcal{P} = (V', \gamma, w_e, C_1, C_2, C_3, C_4, 4),$$

where:

$$
\begin{aligned}
V' &= N \cup T_2 \cup \{c'_1, c'_2, c''_1, c''_2\},\\
\gamma &= (\{1,2,3,4\}, \emptyset),\\
w_e &= h_1(w_1)h_2(w_2),\\
C_1 &= (g'_1(w_1)g'_2(w_2), S_1, R'_1),\\
S_1 &= \{(c'_i; a_i, enter) \mid c'a \to c'v \in R_i, i \in \{1,2\}\}\\
&\quad \cup \{(a'_i, exit) \mid a \in (V - (\{c', c''\} \cup T_1 \cup T_2)), i \in \{1,2\}\}\\
&\quad \cup \{(a, exit) \mid a \in T_2\} \cup \{(\dagger, exit)\},\\
R'_1 &= \{a_i \to t_i(v) \mid c'a \to c'v \in R_i, i \in \{1,2\}\},\\
C_2 &= (g''_1(w_1)g''_2(w_2), S_2, R'_2),\\
S_2 &= \{(c''_i; a_i, enter) \mid c''a \to c''v \in R_i, i \in \{1,2\}\}\\
&\quad \cup \{(a'_i, exit) \mid a \in (V - (\{c', c''\} \cup T_1 \cup T_2)), i \in \{1,2\}\}\\
&\quad \cup \{(a, exit) \mid a \in T_2\} \cup \{(\dagger, exit)\},\\
R'_2 &= \{a_i \to t_i(v) \mid c''a \to c''v \in R_i, i \in \{1,2\}\},\\
C_3 &= \{\lambda, S_3, R_3\}\\
S_3 &= \{(\lambda; a_i, enter) \mid a \to v \in R_i, i \in \{1,2\}\}\\
&\quad \cup \{(a'_i, exit) \mid a \in (V - (\{c', c''\} \cup T_1 \cup T_2)), i \in \{1,2\}\}\\
&\quad \cup \{(a, exit) \mid a \in T_2\} \cup \{(\dagger, exit)\},\\
R_3 &= \{a_i \to t_i(v) \mid a \to v \in R_i, i \in \{1,2\}\},\\
C_4 &= \{\lambda, S_4, R_4\}\\
S_4 &= \{(\lambda; a'_i, enter) \mid a \in (V - (\{c', c''\} \cup T_1 \cup T_2)), i \in \{1,2\}\},\\
&\quad \cup \{(a_i, exit) \mid a \in (V - (\{c', c''\} \cup T_1 \cup T_2)), i \in \{1,2\}\}\\
&\quad \cup \{(\lambda; a, enter) \mid a \in T_2\},\\
R_4 &= \{a'_i \to a_i \mid a \in (V - (\{c', c''\} \cup T_1 \cup T_2)), i \in \{1,2\}\}.
\end{aligned}
$$

The simulation of the P system Π by means of the population P system $\mathcal{P}$ is done in the following way.

Initially, the environment gets assigned the multiset of objects $h_1(w_1)h_2(w_2)$, which contains: an object a_i, with $i \in \{1,2\}$, for each object $a \in (V - \{c', c''\})$ that is assigned to region i in the P system Π and that can evolve by means of some rule in R_i; an object a for each object $a \in (V - \{c', c''\})$ that is assigned to region 2 (the output one) in the P system Π and that cannot evolve by means of any rule in R_2. These latter objects are immediately moved into cell 4 (the output one) during the first step of any computation in $\mathcal{P}$ where they remain as part of the result of the computation in $\mathcal{P}$. Moreover, for each occurrence of the catalyst c' that is present inside region i, with $i \in \{1,2\}$, cell 1 gets assigned an occurrence of the object c'_i; for each occurrence of the catalyst c'' that is present inside region i, with $i \in \{1, 2\}$, cell 2 gets assigned an occurrence of the object c''_i. In this ways, cell 1 and cell 2 are respectively used to simulate the activity of catalyst c', and catalyst c''.

Specifically, for each $i \in \{1, 2\}$, a rule $c'a \to c'v \in R_i$ is simulated in two steps of computations by first using the rule $(c'_i; a_i, enter)$ from S_1 and then the rule $a_i \to t_i(v)$ from R'_1. For each $i \in \{1, 2\}$, a rule $c''a \to c''v \in R_i$ is simulated in two steps of computations by first using the rule $(c''_i; a_i, enter)$ from S_2 and then the rule $a_i \to t_i(v)$ from R'_2 . For each $i \in \{1, 2\}$, a rule $a \to v \in R_i$ is simulated in two steps of computations by first using the rule $(\lambda; a_i, enter)$ from S_3 and then the rule $a_i \to t_i(v)$ from R_3. In this process, the homomorphism t_i, with $i \in \{1, 2\}$, is responsible for assigning to each object produced by a transformation rule the label of the region where the object has to be moved according to the target specified in the corresponding rule in R_i. Furthermore, each object $a \in T_1$ that has to be assigned to region 1 is replaced by the object †; each object $a \in T_2$ that has to be assigned to region 2 is replaced by an object a.

As usual, all these operations are performed in a non-deterministic maximally parallel manner by simulating in this way a step of computation in Π by means of two steps of computation in $\mathcal{P}$. Thus, after having applied some transformation rules in cell 1, cell 2, and in cell 3, these cells always contain some objects of the form a'_i, with $a \in (V - \{c', c''\})$, $i \in \{1, 2\}$, some objects $a \in T_2$, and some occurrences of the object †. All these objects, once produced inside cell 1, cell 2, and cell 3, are immediately released into the environment and then, except for the objects †, they reach cell 4. Each object $a \in T_2$ remains in cell 4 as part of the result of the computation in $\mathcal{P}$ whereas each object of the form a'_i, with $a \in (V - \{c', c''\})$, $i \in \{1, 2\}$, is replaced by the corresponding object a_i, which is eventually released into the environment. In this way, the computation in $\mathcal{P}$ can continue by simulating another step of a computation in Π and so on up until no more rules can be applied to the objects placed inside the cells.

Therefore, the population P system $\mathcal{P}$ correctly simulates the P system Π by generating a set of natural numbers $N(\mathcal{P})$ such that $N(\mathcal{P}) = N(\Pi)$. □

Thus, as an immediate consequence of Theorem 3 and Theorem 8, we obtain the following universality result.

Corollary 9. *$NOPP_{4,1}(R, n\alpha) = NRE$.*

Notice that, from a structural point of view, population P systems of the form considered in Theorem 8 are equivalent to P systems (tissue P systems) with a number of membranes placed at the same level (a number of cells) that are embedded in an unique main membrane (that are connected to an unique distinct cell), which turns to be the environment in population P systems. On the other hand, from a functional point of view, the environment in population P systems is restricted to act as a simple buffer used by the cells to exchange objects.

3.2 The Power of Bond Making

In this section we investigate the power of population P systems where bond making rules are used and where restricted communication rules of the forms $(\lambda; b, in)$, $(\lambda; b, enter)$, $(b, exit)$ are considered. This is to show that bond making rules really increase the power of P systems.

Lemma 10. *(i)* $NOPP_{*,*}(nR, \alpha_0) = NOPP_{1,1}(nR, n\alpha) = NCF$, *and*

(ii) $NET0L \subseteq NOPP_{4,2}(nR, \alpha_1)$.

Proof. *(i)* Consider a population P system $\mathcal{P}$ with $N(\mathcal{P}) \in NOPP_{*,*}(nR, \alpha_0)$. This means all the bond making rules in $\mathcal{P}$ are of the form $(i, \lambda; \lambda, j)$ and the bond making process does not depend on the content of the cells. Thus, as the population of cells is fixed, the structure of the system $\mathcal{P}$ will become completely known after the first step of any computation. In other words, after the first step of any computation, the population P system $\mathcal{P}$ will behave as a population P system that does not use any bond making rule. Therefore, by straightforwardly adapting the construction used in Lemma 6, we can easily construct a population P system with one cell and without bond making rules that is able to simulate the P system $\mathcal{P}$.

(ii) As pointed out in Section 2, for every language $L \in ET0L$ there exists an extended tabled 0L system G with only two tables that generates L, that is, $G = (V, T, w, P_1, P_2)$. Moreover, after having used the table P_1, we can use both P_1 and P_2 but, after having used the table P_2, we always use the table P_1. The table used in the first step of a computation is P_1. Thus, in order to simulate the L system G, we construct the following population P system

$$\mathcal{P} = (V', \gamma, \alpha, w_e, C_1, C_2, C_3, C_4, 4),$$

where:

$$\begin{aligned}
V' &= \{a, a' \mid a \in V\} \cup \{p_1, p_1', p_2, p_2', f\},\\
\gamma &= (\{1, 2, 3, 4\}, \emptyset),\\
\alpha &= \{(3, p_1; \lambda, 1), (3, p_2; \lambda, 2), (3, f; \lambda, 4), (1, p_1'; \lambda, 3), (2, p_2'; \lambda, 3)\},\\
w_e &= \lambda,\\
C_1 &= (\lambda, S_1, R_1),\\
S_1 &= \{(\lambda; a', in) \mid a \in V\} \cup \{(\lambda; p_1, in)\},\\
R_1 &= \{a' \to v \mid a \to v \in P_1\} \cup \{p_1 \to p_1'\},
\end{aligned}$$

$C_2 = (\lambda, S_2, R_2),$
$S_2 = \{(\lambda; a', in) \mid a \in V\} \cup \{(\lambda; p_2, in)\},$
$R_2 = \{a' \to v \mid a \to v \in P_2\} \cup \{p_2 \to p_2'\},$
$C_3 = (w\,p_2', S_3, R_3),$
$S_3 = \{(\lambda; a, in) \mid a \in V\} \cup \{(\lambda; p_1', in), (\lambda; p_2', in)\},$
$R_3 = \{a \to a' \mid a \in V\} \cup \{p_1' \to p_1, p_1' \to p_2, p_2' \to p_1, p_1' \to f, p_2' \to f\},$
$C_4 = (\lambda, S_4, R_4),$
$S_4 = \{(\lambda; a', in) \mid a \in V\},$
$R_4 = \{a' \to a' \mid a \in (V - T)\}.$

The population P system $\mathcal{P}$ simulates the L system G in the following way.

The axiom w is initially placed inside cell 3 together with the object p_2'. The use of the table P_1 is then simulated by applying the rule $p_2' \to p_1$ from R_3, and a rule $a \to a'$ from R_3, for each symbol $a \in V$ contained in cell 3. Now, the presence of the object p_1 makes possible to form a bond between cell 3 and cell 1, which allows all the object a', with $a \in V$, to be moved from cell 3 into cell 1 together with the object p_1. In cell 1, the simulation of the table P_1 is completed by applying in a maximal parallel manner all the rules $a' \to v$ from R_1, with $a \to v$ a rule in P_1. As well as this, we apply the rule $p_1 \to p_1'$. Next, a bond between cell 1 and cell 3 is created, which allows all the objects $a \in V$ in cell 1 to get back to cell 3 together with the object p_1'. At this point, the simulation of the L system can continue by applying either the rule $p_1' \to p_1$ from R_3 (i.e., we want to use the table P_1), or the rule $p_1' \to p_2$ from R_3 (i.e., we want to use the table P_2). The use of the table P_2 is simulated in a very similar way by using cell 2 instead of cell 1. In particular, when the simulation of P_2 is finished, an object p_2' is produced inside cell 3 in order to specify that, after having used P_2, we have to use the table P_1.

At any time, we can stop simulating tables in G by applying either the rule $p_1' \to f$ from R_3, or the rule $p_2' \to f$ from R_3 depending on the table that has been used in the previous step of computation. Anyway, in the presence of an object f, a bond between cell 3 and cell 4 (the output one) is created, which allows all the objects a' placed inside cell 3 to be moved from cell 3 into cell 4. Now, the computation can halt if and only if cell 4 does not contain any non-terminal object. This is because of the presence of a rule $a' \to a'$, for each $a \in (V - T)$.

In conclusion, the population P system $\mathcal{P}$ correctly simulates the L system G. □

Therefore, population P systems that use bond making rules are able to generate (at least) the whole family of length sets of extended tabled 0L systems. This means they are able to generate non-semilinear sets of natural numbers, and such sets are not in the family of length sets of context-free languages. Next, we provide an universality result for population P systems that use bond making rules of size two by showing they are able to simulate counter machines. On the other hand, finding an upper bound for the power of population P systems with bond making rules of size at most one remains an open problem.

Theorem 11. *$NOPP_{6,2}(nR, \alpha_2) = NRE$.*

Proof. Let $M = (Q, F, p_0, \{a_1, a_2, a_3\}, a_3, I)$ be a three counter machine as specified in Definition 1.

In order to simulate the counter machine M, we construct a population P system $\mathcal{P}$ such that

$$\mathcal{P} = (V, \gamma, \alpha, w_e, C_1, C_2, C_3, C_4, C_5, C_6, 3),$$

where:

$$\begin{aligned}
V &= \{q, q', q'' \mid q \in q\} \cup \{a, \bar{a} \mid a \in A\} \cup \{d_{a,q}, d'_{a,q}, e_{a,q}, e'_{a,q} \mid a \in A, q \in Q\} \\
&\quad \cup \{d_j, e_j \mid 1 \le j \le 5\} \cup \{d, e, f, \#, \#_1\}, \\
\gamma &= (\{1, 2, 3, 4, 5, 6\}, \emptyset), \\
\alpha &= \{(4, d_{a_i,q}; a_i, i) \mid 1 \le i \le 3, q \in Q\} \cup \{(4, \bar{a}\bar{a}; \lambda, 5), (4, \bar{a}; d_5, 6) \mid a \in A\} \\
&\quad \cup \{(4, e'_{a_i,q}; a_i, i), (4, e'_{a_i,q}; e_5, 6) \mid 1 \le i \le 3, q \in Q\} \cup \{(6, f; \lambda, 5)\}, \\
w_e &= \lambda, \\
C_i &= (\lambda, \{(\lambda; a_i, enter), (\lambda; e'_{a_i,q}, in)\}, \{e'_{a_i,q} \to \#, \# \to \#\}), \\
&\qquad \text{for each } 1 \le i \le 3, \\
C_4 &= (\#, S_4, R_4), \\
S_4 &= \{(a, exit), (\lambda; a, in), (\lambda; d_{a,q}, enter), (\lambda; e_{a,q}, enter) \mid a \in A, q \in Q\}, \\
R_4 &= \{d_{a,q} \to d'_{a,q}, e_{a,q} \to e'_{a,q}, a \to \bar{a} \mid a \in A, q \in Q\}, \\
C_5 &= (\lambda, S_5, R_5), \\
S_5 &= \{(\lambda; \#, in), (\lambda; \#_1, in), (\lambda; f, in)\}, \\
R_5 &= \{\# \to \#\}, \\
C_6 &= (p_0\, \#_1, S_6, R_6), \\
S_6 &= \{(a, exit), (\bar{a}, exit), (\lambda; \bar{a}, in), (d_{a,q}, exit), (\lambda; d'_{a,q}, in) \mid a \in A, q \in Q\} \\
&\quad \cup \{(e_{a,q}, exit), (\lambda; e'_{a,q}, in) \mid a \in A, q \in Q\} \cup \{(d_5, exit), (e_5, exit)\}, \\
R_6 &= \{p \to q''a \mid (p \to q, +a) \in I\} \cup \{p \to d\, d_{a,q} \mid (p \to q, -a) \in I\} \\
&\quad \cup \{p \to q \mid (p \to q, \epsilon) \in I\} \cup \{p \to e\, e_{a,q} \mid (p \to q, a = 0) \in I\} \\
&\quad \cup \{q'' \to q', q' \to q \mid q \in Q\} \cup \{q \to f \mid q \in F\} \\
&\quad \cup \{d'_{a,q} \to q, e'_{a,q} \to q \mid a \in A, q \in Q\} \\
&\quad \cup \{d_j \to d_{j+1}, e_j \to e_{j+1} \mid 1 \le j \le 4\} \cup \{d \to d_1, e \to e_1, \#_1 \to \#_1\}.
\end{aligned}$$

Now, after some steps of computation, we can suppose the population P system $\mathcal{P}$ to have reached a configuration of the form:

$$\langle(\{1,2,3,4,5,6\},\emptyset), w_\dagger, w_1, w_2, w_3, w_4, w_5, w_6\rangle, \quad (1)$$

with $w_\dagger \in V^*$, a multiset of objects that are in the environment and that cannot enter the cells anymore, $w_i \in \{a_i\}^*$, for each $1 \leq i \leq 3$, $|w_i|$ the current value of the counter $a_i \in A$, $w_4 = \#, w_5 = \lambda, w_6 = p\#_1$, and $p \in Q$, the current state of the counter machine M. Initially, we have $w_\dagger = \lambda$, $w_1 = \lambda$, $w_2 = \lambda$, $w_3 = \lambda$, and $p = p_0$. Thus, by starting from a configuration like (1), the simulation of the instructions in I is done in the following way.

Simulation of an instruction $(p \to q, +a) \in I$. In cell 6, we apply the rule $p \to q''a$, which produces an object q'', with $q \in Q$, and an object $a \in A$. In two further steps of computation, the object a reaches the cell i such that $a = a_i$, $1 \leq i \leq 3$, by passing through the environment. At the same time, the new state q is produced inside cell 6 by applying in sequence the rules $q'' \to q'$, $q' \to q$ from R_6. In this way, we add 1 to the current value of the counter a_i and we move the machine from state p to state q.

Simulation of an instruction $(p \to q, \epsilon) \in I$. We just apply the rule $p \to q$ from R_6 in order to produce the new state q inside cell 6.

Simulation of an instruction $(p \to q, -a) \in I$. In cell 6, we apply the rule $p \to d\, d_{a,q}$, which produces an object d and an object $d_{a,q}$, with $a \in A$, $q \in Q$. In two further steps of computation, the object $d_{a,q}$ reaches cell 4 by passing through the environment. At the same time, the object d_2 is produced inside cell 6. In cell 4, the object $d_{a,q}$ is used to check whether the cell i such that $a = a_i$, $1 \leq i \leq 3$, contains some objects a_i or not (i.e., to check whether the current value of the counter a_i is greater than or equal to 1 or not). In the former case, a bond between cell 4 and cell i can be created by using the rule $(4, d_{a_i,q}; a_i, i)$ from α, and the simulation of the current instruction can continue; in the other case, this bond cannot be created and the simulation of the current instruction will stop after having produced the object d_5 inside cell 6. Nevertheless, the computation in $\mathcal{P}$ will not halt because of the rule $\#_1 \to \#_1$ in R_6.

Thus, after having produced a bond between cell 4 and cell i, the whole content of cell i is moved from cell i to cell 4 by applying in a maximally parallel manner the rule $(\lambda; a_i, in)$ from S_4. At the same time, the object d_2 is replaced by d_3 inside cell 6 while the object $d_{a_i,q}$ is replaced by $d'_{a_i,q}$ inside cell 4. Now, without loss of generality, we assume $i = 1$ as the simulation proceeds exactly in the same way irrespectively of the value of i.

Next, the object d_3 is replaced by d_4 in cell 6 while, for each object a_1 in cell 4, we can use either a rule $(a_1, exit) \in S_4$ or a rule $a_1 \to \bar{a}_1 \in R_4$. This means the system is able to reach a configuration of the form

$$\langle(\{1,2,3,4,5,6\},\emptyset), w_\dagger\, w_1', \lambda, w_2, w_3, \bar{w}_1 d'_{a_1,q}\#, \lambda, d_4\#_1\rangle, \qquad (2)$$

with $w_1' \in \{a_1\}^*$, $\bar{w}_1 \in \{\bar{a}_1\}^*$, and $|w_1'| + |\bar{w}_1| = |w_1|$. Now, we show that the simulation of the instruction $(p \to q, -a) \in I$ can be successfully completed if and only if $|\bar{w}_1| = 1$ (i.e., we have subtracted 1 from the current value of the counter a_1). To this aim, we consider the following cases.

1. $|\bar{w}_1| = 0$. The simulation of the current instruction will stop after having produced d_5 inside cell 6 and having returned w_1' in cell 1, but the computation will not halt because of the rule $\#_1 \to \#_1$ in R_6.

2. $|\bar{w}_1| \geq 2$. This means a bond between cell 4 and cell 5 is created by using the bond making rule $(4, \bar{a}_1\bar{a}_1; \lambda, 5) \in \alpha$. Then, in the next step, the object $\#$ is moved from cell 4 to cell 5 where an infinite computation is generated by means of the rule $\# \to \# \in R_5$.

3. $|\bar{w}_1| = 1$. No bonds between cell 4 and cell 5 can be created and the simulation of the current instruction continues by producing the object d_5 inside cell 6 and by moving the multiset w_1' from the environment into cell 1. Next, a bond is created between cell 4 and cell 6 by using the rule $(4, \bar{a}_1; d_5, 6) \in \alpha$, which makes possible to move the objects $d'_{a_1,q}$, $\bar{a_1}$ from cell 4 to cell 6. In the meanwhile, the object d_5 is released in the environment and it will not be able to enter the cells anymore. Finally, in cell 6, the object $d'_{a_1,q}$ is used to produce the new state q while the object $\bar{a_1}$ is removed from cell 6 by releasing it in the environment. In this way, we obtain a configuration like (1) where the multiset w_1 is replaced by a multiset w_1' such that $|w_1'| = |w_1| - 1$.

Simulation of an instruction $(p \to q, a = 0) \in I$. In cell 6, we apply the rule $p \to e\, e_{a,q}$, which produces an object e and an object $e_{a,q}$, with $a \in A$, $q \in Q$. In two further steps of computation, the object $e_{a,q}$ reaches cell 4 by passing through the environment. In cell 4, the object $e_{a,q}$ is then replaced by $e'_{a,q}$. At the same time, the object e_3 is produced inside cell 6. Next, the object $e'_{a,q}$ is used to check whether the cell i such that $a = a_i$, $1 \leq i \leq 3$, contains some objects a_i or not (i.e., to check whether the current value of the counter a_i is greater than or equal to 1 or not). In the former case, a bond between cell 4 and cell i can be created by using the rule $(4, e'_{a_i,q}; a_i, i)$ from α, which makes possible to move the object $e'_{a,q}$ from cell 4 to cell i. As a consequence of this fact, an infinite computation is generated because of the rules $e'_{a_i,q} \to \#$, $\# \to \#$ in R_i. Otherwise, if cell i does not contain any object a_i (i.e., the current value of the counter a_i is 0), the system is able to reach a configuration where cell 6 contains an object

e_5 and cell 4 still contains the object $e'_{a,q}$. At that point, a bond between cell 4 and cell 6 is created by using the bond making rule $(4, e'_{a_i,q}; e_5, 6)$, which allows the object $e'_{a,q}$ to be moved from cell 4 to cell 6 while the object e_5 is released into the environment. Finally, in cell 6, the object $e'_{a,q}$ is replaced by the state q and this completes the simulation of the instruction $(p \rightarrow q, a = 0) \in I$.

At any moment, when a final state $p \in F$ is produced inside cell 6, we have to apply a rule $p \rightarrow f$ from R_6, which makes possible to form a bond between cell 6 and cell 5. Thus, the object f can be moved from cell 6 to cell 5 together with the object $\#_1$, and the computation halts by having correctly simulated a sequence of instructions in M.

Therefore, as a consequence of the universality of three counter machines, we obtain immediately the universality of population P systems with bond making rules of size at most 2. □

4 Population P systems with Active Cells

The model of population P systems is now augmented with an operation of cell division as a mechanism to introduce new cells in the system, and with an operation of cell death as a mechanism to remove cells from the system. As well as this, an operation of cell differentiation is considered that allows the types of the cells to be changed by varying in this way the sets of rules that can be used inside the cells.

Definition 12. A population P system with active cells is a construct

$$\mathcal{P} = (V, K, \gamma, \alpha, w_e, C_1, C_2, \ldots, C_n, R),$$

where:

1. V is a finite alphabet of symbols called objects;
2. K is a finite alphabet of symbols, which define different types for the cells;
3. $\gamma = (\{1, 2, \ldots, n\}, E)$, with $E \subseteq \{\{i, j\} \mid 1 \leq i \neq j \leq n\}$, is a finite undirected graph;
4. α is a finite set of bond making rules $(t, x_1; x_2, p)$, with $x_1, x_2 \in V^*$, and $t, p \in K$;
5. $w_e \in V^*$ is a finite multiset of objects initially assigned to the environment;
6. $C_i = (w_i, t_i)$, for each $1 \leq i \leq n$, with $w_i \in V^*$ a finite multiset of objects, and $t_i \in K$ the type of cell i;
7. R is a finite set of rules of the forms:

(a) $(a; b, in)_t$, $(a; b, enter)_t$, $(b, exit)_t$, for $a \in V \cup \{\lambda\}$, $b \in V$, $t \in K$ (*communication rules* that allows a cell to exchange with its neighbouring cells or the environment according to the cell type and the existing bonds among the cells; the meaning of these rules is exactly the same as the meaning of communication rules considered in Definition 5),

(b) $(\, a \to y\,)_t$, for $a \in V$, $y \in V^+$, $t \in K$ (*transformation rules*: an object a can be replaced by a non-empty multiset y inside a cell of type t),

(c) $(\, a\,)_t \to (\, b\,)_p$, with $a, b \in V$, $t, p \in K$ (*cell differentiation rules*: an object a can be used inside a cell of type t to change the type of the cell from t to p; as a consequence of this operation, the object a is replaced by an object b),

(d) $(\, a\,)_t \to (\, b\,)_t\,(\, c\,)_t$, with $a, b, c \in V$, $t \in K$ (*cell division rules*: an object a can be used inside a cell of type t to produce a new cell of type t, which contains the same objects as the originating one except for the object a that is replaced by c; the originating cell is retained with the same content except for the object a that is replaced by b),

(e) $(\, a\,)_t \to \dagger$, with a, $t \in K$ (*cell death rule*: an object a can be used inside a cell of type t to cause the removal of the cell from the system; as a consequence of this operation, the objects contained in the dying cell are removed from the system).

Now, with respect to Definition 5, a population P system with active cells $\mathcal{P}$ is defined as a finite collection of $n \geq 1$ cells where each cell is characterized by a multiset of objects defining its initial content and a symbol from a distinct alphabet K, which defines the type of the cell. At any time, many different cells of the same type can be present in the population system $\mathcal{P}$. All these cells evolve according to the rules in R, which allow them to exchange objects with the environment or with their neighbouring cells (*communication rules*), to modify the objects defining their content (*transformation rules*), to change the type they have got assigned (*cell differentiation rules*), to divide by introducing new cells in the system (*cell division rules*), and to be removed from the system (*cell death rule*). In particular, for each cell that is present in the system, the set of rules that can be used depends on the type the cell has got assigned. Moreover, as in Definition 5, the structure of the system is given by a finite directed graph where the nodes correspond in an one-to-one manner to the cells in the system and the edges define the bonds that exist among these cells. Yet again, these edges influence cell capability of achieving direct communication and they can be modified by the bond making rules in α.

A configuration of a population P system with active cells $\mathcal{P}$ at any time is given by a tuple Σ such that:

$$\Sigma = \langle \gamma', w'_e, C'_1, C'_2, \dots, C'_n \rangle,$$

with $\gamma' = (\{1, 2, \dots, n\}, E')$, for $E' \subseteq \{\{i, j\} \mid 1 \leq i \neq j \leq n\}$ (the graph that defines the current structure of the system $\mathcal{P}$), $w'_e \in V^*$ (the multiset of objects that are currently associated with the environment) and, for all $1 \leq i \leq n$, $C'_i = (w'_i, t_i)$, for $w'_i \in V^*$ (the multiset of objects that defines the current content of cell i) , $t_i \in K$ (the current type of cell i). Then, in a similar way to Section 3, we define a single step of a computation in $\mathcal{P}$ (denoted by $\Longrightarrow_{\mathcal{P}}$) as being formed by two separate stages: a stage of transformation-communication (denoted by $\stackrel{tc}{\Longrightarrow}_{\mathcal{P}}$) and a stage of bond making (denoted by $\stackrel{bm}{\Longrightarrow}_{\mathcal{P}}$).

During the stage of transformation-communication, rules of the forms (a), (b) are applied to the objects contained in the cells in a non-deterministic maximally parallel manner but with the restriction that at most one rule of the form (c), (d), or (e) per each cell can be used at a time that is non-deterministically chosen. In this respect, we assume the objects to be first communicated/modified by means of rules of the forms (a), (b) and the population of cells to be then modified by means of rules of the forms (c), (d), or (e). However, objects are assigned to rules of the forms (c), (d), (e) before any communication/modification of these takes place. Specifically, if a cell $C'_i = (w''_i, t_i)$ evolves by means of a rule $(\,a\,)_t \rightarrow (\,b\,)_p$, then it is replaced by a cell $C''_i = (w''_i, p_i)$ where w''_i is the multiset obtained by replacing in w'_i an occurrence of a with an occurrence of b. The structure of the system is not changed by this particular operation as the new cell C''_i is associated with the same node i in the graph defining the structure of the system that was previously associated with cell C'_i. If a cell $C'_i = (w''_i, t_i)$ evolves by means of a rule $(\,a\,)_t \rightarrow (\,b\,)_t\,(\,c\,)_t$, then it is replaced by two new cells $C''_i = (w''_i, t_i)$, and $C''_{i'} = (w''_{i'}, t_{i'})$ where w''_i is the multiset obtained by replacing in w'_i an occurrence of a with an occurrence of b, and $w''_{i'}$ is the multiset obtained by replacing in w'_i an occurrence of a with an occurrence of c. As a consequence of this particular operation, cell C''_i is associated with the same node i that was previously associated with cell C'_i in the graph defining the structure of the system, while cell $C''_{i'}$ is associated with a new node i' that is now added to the graph defining the structure of the system without being connected to any other node in that graph. If a cell $C'_i = (w''_i, t_i)$ evolves by means of a rule $(\,a\,)_t \rightarrow \dagger$, then it is removed from the system together with the corresponding node i in the graph defining the structure of the system and all the edges of the form $\{i, j\}$, with $i \neq j$.

During the stage of bond making, the set of edges in the graph defining the structure of the system is modified by using the bond making rules in α. Obviously, we have here to consider the graph that results from the application of rules of the forms (c), (d), (e) in the previous transformation-communication stage. Specifically, an edge $\{i, j\}$, with $i \neq j$, is added to the graph defining the structure of the system if there exists a bond making rule $(t, x_1; x_2, p) \in \alpha$, with t the type of cell i, x_1 a multiset that is contained in cell i, x_2 a multiset that is contained in cell j, and p the type of cell j.

Finally, after this process of bond making, we assume the nodes in the graph to be labelled in a one-to-one manner with values in $\{1, 2, \dots, m\}$, for $m \geq 0$ the number of cells in the system.

A successful computation in $\mathcal{P}$ is then defined as a finite sequence of transitions of the form

$$\Sigma_0 \Longrightarrow_{\mathcal{P}} \Sigma_1 \Longrightarrow_{\mathcal{P}} \dots \Longrightarrow_{\mathcal{P}} \Sigma_{k-1} \Longrightarrow_{\mathcal{P}} \Sigma_k,$$

where $k \geq 1$, Σ_0 is the initial configuration of the P system $\mathcal{P}$ as specified in Definition 12, and Σ_k is a final configuration such that there does not exist any configuration $\Sigma' \neq \Sigma_k$, and $\Sigma_k \stackrel{tc}{\Longrightarrow}_{\mathcal{P}} \Sigma'$. In other words, a successful computation is a computation that halts in a configuration where, after a last bond making stage, the current population of cells in the system cannot be modified anymore by means of any rule. This time the result of a computation is given by the number of objects that are associated with the environment in the final configuration Σ_k. The set of natural numbers that are generated in this way by all the successful computations in $\mathcal{P}$ is denoted by $N(\Pi)$.

At this point, we can introduce the families of sets of natural numbers $NOPP_{n,k}(t_m, o, b)$, with $n \geq k \geq 1, m \geq 1, o \subseteq \{a, b, c, d, e\}$ or $o \subseteq \{a', b, c, d, e\}$, and $b \in \{n\alpha, \alpha_t \mid t \geq 0\}$, which are generated by population P systems where:

- in each step of a computation, the number of cells in the system is always less than or equal to n;
- in each step of a computation, the number of cells in each connected component of the graph defining the structure of the system is always less than or equal to k;
- the number of different types that are considered for the cells is at most m;
- rules of the forms indicated in o are considered; for rules of the form (a) we consider also the case of restricted communication rules denoted by a', which corresponds to the case nR considered in the previous sections;
- b specifies the kind of bond making rules that can be used inside the system as described in Section 3.

4.1 The Power of Cell Division/Differentiation

Here, we provide some results that concern the computational power of populations P systems with active cells.

Lemma 13. $NET0L \subseteq NOPP_{1,1}(t_3, \{a', b, c\}, n\alpha)$.

Proof. Consider an extended tabled 0L system G with only two tables, that is, $G = (V, T, w, P_1, P_2)$ such that, after having used the table P_1, we can use both P_1 and P_2 but, after having used the table P_2, we always use the table P_1. Moreover, the table used in the first step of a computation is table P_1. As already mentioned before, extended tabled 0L systems of this form are able to generate the whole family of length sets of extended tabled 0L system.

Thus, we construct a population P system with active cells $\mathcal{P}$ that simulates the L system G such that

$$\mathcal{P} = (V', \{p_1, p_2, f\}, (\{1\}, \emptyset), \emptyset, (w\, p_1, p_1), R)$$

with $V' = V \cup T \cup \{p_1, p_2, f\}$, and R is a finite set of rules that contains: a rule $(\, a \to v\,)_{p_i}$, for each $a \to v \in P_i$, $1 \leq i \leq 2$, the rules $(\, p_1\,)_{p_1} \to (\, p_1\,)_{p_1}$, $(\, p_1\,)_{p_1} \to (\, p_2\,)_{p_2}$, $(\, p_2\,)_{p_2} \to (\, p_1\,)_{p_1}$, $(\, p_1\,)_{p_1} \to (\, f\,)_f$, $(\, p_2\,)_{p_2} \to (\, f\,)_f$, a rule $(\, a \to a\,)_f$, for each $a \in V - T$, and a rule $(a, exit)_f$, for each $a \in V$.

Now, it is easy to see that the population P system $\mathcal{P}$ correctly simulates the L system G. In fact, at any moment, the type of cell 1 corresponds to the table of rules that must be applied in the next step of a computation; each rule in the table is then simulated by applying the corresponding transformation rules inside cell 1 in a non-deterministic maximally parallel manner. As well as this, we always apply a cell differentiation rule that changes the type of cell 1 in such a way to pass to another table of the L system G. Moreover, at any moment, we can decide to finish a computation by applying a cell differentiation rule that assigns the type f to cell 1. In that case, if cell 1 contains only terminal symbols, then the computation halts by releasing all these objects in the environment; otherwise, an infinite computation is generated because of the rule $(\, a \to a\,)_f$, with $a \in V - T$. □

The previous result shows that population P systems with active cells are quite powerful computational devices. In fact, population P systems with one cell equipped with the sole operation of cell differentiation are able to generate (at least) the whole family of length sets of tabled extended 0L systems, which contains non-semilinear sets of natural numbers. Next, we show that population P systems of this form turn to be computationally complete once at least two cells are considered.

Theorem 14. $NOPP_{2,1}(t_*, \{a', b, c\}, n\alpha\} = NRE$.

Proof. This universality result is again obtained by simulating a counter machines. Let $M = (Q, F, p_0, A, I)$ be a three counter machine as specified in Definition 1. We construct the following P system with active cells that is able to simulate the counter machines M

$$\mathcal{P} = (V, K, (\{1,2\}, \emptyset), \lambda, (p_0\,\#_1 \to \#_1, 0), (\lambda, 1), R).$$

In this case we do not enter the details of the formalisation of the P system $\mathcal{P}$ but we just provide the rules that are necessary to simulate each instruction of the machine M.

Simulation of an instruction $(p \to q, +a) \in I$. We use the following rules in order to simulate an instruction of this form.

1. $(\,p \to q''\,a\,)_0$,
2. $(a, exit)_0$,
3. $(\lambda; a, enter)_1$,
4. $(\,q'' \to q'\,)_0$,
5. $(\,q' \to q\,)_0$.

In this way, we add an object a to the content of cell 2 of type 1.

Simulation of an instruction $(p \to q, \epsilon) \in I$. We just need a rule $(\,p \to q\,)_0$ in order to simulate an instruction of this form.

Simulation of an instruction $(p \to q, -a) \in I$. We use the following rules in order to simulate an instruction of this form.

1. $(\,p \to a_q\,)_0$,
2. $(a_q, exit)_0$,
3. $(\lambda; a_q, enter)_1$,
4. $(\,a_q\,)_1 \to (\,\$\,)_{a_q}$,
5. $(\,a\,)_{a_q} \to (\,q\,)_1$,
6. $(q, exit)_1$,
7. $(\lambda; q, enter)_0$,
8. $(\,\#_1 \to \#_1\,)_0$.

In this case, when the object a_q enters cell 2 of type 1, the type of this cell is changed into a_q and a dummy object \$ is produced, which cannot be used in any rule in the system. Next, if cell 2 contains an object a, then the simulation of the current instruction can be completed by first applying the rule $(\,a\,)_{a_q} \to (\,q\,)_1$, then the rule $(q, exit)_1$, and finally the rule $(\lambda; q, enter)_0$. Instead, if cell 2 does not contain any object a, the simulation of the current instruction cannot be completed and an infinite computation is generated because of the rule $(\,\#_1 \to \#_1\,)_0$.

Simulation of an instruction $(p \to q, a = 0) \in I$. We use the following rules in order to simulate an instruction of this form.

1. $(\,p \to a'_q\, d\,)_0$,
2. $(a'_q, exit)_0$,
3. $(d, exit)_0$,
4. $(\lambda; a'_q, enter)_1$,
5. $(\lambda; d, enter)_1$,
6. $(\,d \to d'\,)_1$,
7. $(\,a'_q\,)_1 \to (\,\$\,)_{a'_q}$,
8. $(\,d' \to d''\,)_{a'_q}$,
9. $(\,a\,)_{a'_q} \to (\,\#\,)_{\#}$,
10. $(\,d''\,)_{a'_q} \to (\,q\,)_1$
11. $(\,\# \to \#\,)_{\#}$,
12. $(q, exit)_1$,
13. $(\lambda; q, enter)_0$,

In this case, when the objects a'_q, d enter cell 2 of type 1, the type of this cell is changed into a'_q while the object d is replaced by d'. Next, if cell 2 contains some objects a, then an infinite computation is generated because of the rules $(\,a\,)_{a'_q} \to (\,\#\,)_{\#}, (\,\# \to \#\,)_{\#}$. Instead, if cell 2 does not contain any object a, then the simulation of the current instruction can be completed by first applying the rule $(\,d''\,)_{a'_q} \to (\,q\,)_1$, then the rule $(q, exit)_1$, and finally the rule $(\lambda; q, enter)_0$.

Finally, we need the following rules in order to produce the output in the environment.

1. $(\,p\,)_0 \to (\,f\,)_f$, with $p \in F$,
2. $(f, exit)$,
3. $(\lambda; f, enter)_1$,

4. $(\,f\,)_1 \to (\,f\,)_f$,
5. $(a, exit)_f$.

In particular, the first rule is used to stop cell 1 using the rule $(\,\#_1 \to \#_1\,)_0$. $\square$

Finally, we show the universality of population P systems that use the operations of cell division and cell death in combination with bond making rules of size at most 1 where the types of the cells are never changed.

Theorem 15. $NOPP_{7,2}(t_6, \{a, b, d, e\}, \alpha_1) = NRE$.

Proof. In a sense, this result is a consequence of Theorem 11 that provides the universality of population P systems with bond making rules of size two. In fact, in the proof of Theorem 11, a bond making rule of size two is used only in the process of simulating instructions of the form $(p \to q, -a)$ in a counter machine M. This is necessary in order to control the parallelism and make sure that just one object a is removed from the cell containing the current value of the counter a. Here, the control on the parallelism can be achieved by using the operation of cell division that allows a cell to pick up a single occurrence of an object to be used for cell division purposes. Furthermore, with respect to Theorem 11, we need now to produce the output as a multiset of objects that is associated with the environment in the final configuration rather than with a specific output cell. Therefore, let $\mathcal{P}$ be the population P system used in the proof of Theorem 11. We construct a population P system with active cell $\mathcal{P}'$ with the same initial configuration in terms of cells and multisets of objects associated with the cells. Specifically, each cell i in $\mathcal{P}$, $1 \le i \le 6$, gets assigned i itself as type of cell i. Moreover, cell 5 in $\mathcal{P}'$ is used to produce the output at the end of a successful computation by collecting the objects that are contained in cell 3 (the output one in $\mathcal{P}$) and releasing them in the environment in such a form that cannot be rentered in any cell anymore.

However, we do not enter here in all the details of the population P system $\mathcal{P}'$ but we just show how to simulate instructions of the form $(p \to q, -a)$ in a given counter machine M by using cell division and bond making rules of size 1. The simulation of the other instructions in the machine M is done exactly in the same way as in the population P system $\mathcal{P}$ of Theorem 11.

- $(\lambda; a_i, enter)_i$, for each $1 \le i \le 3$; each cell i of type i in $\mathcal{P}'$ is used to hold the current value of the counter a_i and when objects of the form a_i are produced in the environment they have to enter cell i.
- $(a, exit)_4$, $(\lambda; a, in)_4$, $(\lambda; d_{a,q}, enter)_4$, $(\,d_{a,q} \to d'_{a,q}\,)_4$, $(\,a\,)_4 \to (\,a'\,)_4(\,\bar{a}\,)_4$, $(\,a'\,)_4 \to \dagger$, with a a counter in the machine M and q a state of the machine M; these rules correspond to the rule used in cell 4 of the system $\mathcal{P}$, which are now augmented with a rule of cell division and a rule of cell death.

– $(a, exit)_6$, $(d_{a,q}, exit)_6$, $(\lambda; \bar{a}, in)_6$, $(\lambda; d'_{a,q}, in)_6$, $(\,\bar{a} \to \$\,)_6$, $(\,d_5 \to \$\,)_6$, $(\,p \to d\,d_{a,q}\,)_6$, $(\,d'_{a,q} \to q\,)_6$, $(\,d \to d_1\,)_6$, $(\,d_j \to d_{j+1}\,)_6$, $(\,\#_1 \to \#_1\,)$, with a a counter in the machine M, q a state of the machine M, $\$, d$ two new symbols, and $1 \leq j \leq 4$; these rules correspond to the rules used in cell 6 in the system $\mathcal{P}$; here, communication rules that are used to remove objects from the system by releasing them in the environment are replaced by transformation rules that produce the dummy object \$.

As well as this, we consider the bond making rules $(4, d_{a_i,q}; a_i, i)$, with $1 \leq i \leq 3$, and $(4, \bar{a}; d_5, 6)$.

Now, as in the proof of Theorem 11, consider a configuration of the form

$$\langle(\{1,2,3,4,5,6\}, \emptyset), \lambda, w_1, w_2, w_3, \lambda, \lambda, p\#_1 w_\$\rangle, \tag{3}$$

with $w_i \in \{a_i\}^*$, for each $1 \leq i \leq 3$, $|w_i|$ the current value of the counter $a_i \in A$, $p \in Q$, the current state of the counter machine M, and $w_\$ \in \{\$\}^*$.

In order to simulate an instruction of the form $(p \to q, a-)$, in the cell of type 6, we apply the rule $(\,p \to d\,d_{a,q}\,)_6$, which produces an object d and an object $d_{a,q}$. In two further steps of computation, the object $d_{a,q}$ reaches cell 4 (of type 4) by passing through the environment. At the same time, the object d_2 is produced inside cell 6 (of type 6). In cell 4, as in the proof of Theorem 11, the object $d_{a,q}$ is used to create a bond between cell 4 and the cell i such that $a = a_i$, $1 \leq i \leq 3$. This makes possible to move the whole content of cell i into cell 4. Next, for each object a that has been moved inside cell 4, we apply a rule $(a, exit)_4$ in parallel with a cell division rule $(\,a\,)_4 \to (\,a'\,)_4(\,\bar{a}\,)_4$, which can be used by just one occurrence of the object a. This means we always produce in the environment a multiset w'_1 such that $|w'_1| = |w_1| - 1$, which is eventually returned to cell i. At the same time, we obtain two cells of type 4 that respectively contain an object a' and an object $\bar{a}$. The cell that contains the object a' is immediately removed from the system by using the cell death rule $(\,a'\,)_4 \to \dagger$ while the object $\bar{a}$ in the remaining cell of type 4 is used to form a bond with cell 6 when the object d_5 is produced inside that cell. In this way, we can complete the simulation of the instruction $(p \to q, a-)$ by producing the new state q inside cell 6 as illustrated in the proof of Theorem 11. □

5 Conclusions

In this paper we have introduced a notion of population P systems as a class of tissue P systems where the links between the cells can be modified by means of a specific set of bond making rules. As well as this, cell division rules which introduce new cells into the system, cell differentiation rules which change the set of rules that can be used inside of a cell, and cell death rules which remove cells from the system are also considered by introducing a particular notion of population P systems with active cells. The paper mainly reports universality results for the following models:

1. unstructured population P systems where cells are restricted to communicate only by means of the environment and being equipped with communication rules of the form introduced in [Bernardini and Gheorghe 2004b] (Corollary 9);
2. population P systems equipped with bond making rules of size at most two and simple communication rules where objects can be moved from a cell to another one without any control mechanism (Theorem 11);
3. unstructured population P systems where cells are restricted to communicate only by means of the environment equipped with the sole operation of cell differentiation (Theorem 14);
4. population P systems equipped with bond making rules of size at most one and the operation of cell division (Theorem 15).

These last two results are of particular interests if compared with similar results obtained for P systems with active membranes [Păun 2002]. In fact, Theorem 14 proves the universality for P systems that use an operation for changing the labels of the cells (i.e., the types of the cells) when communication rules are applied in a maximally parallel manner and not in a sequential manner as in P systems with active membranes [Păun 2002]. Then, Theorem 15 povides an universality result for systems where the labels of the cells (i.e., the types of the cells) are never changed by just considering a separate mechanism for modifying the bonds among the cells. However, alongside this direction of comparing population P systems and P systems with active membranes, it remains to be investigated the efficiency of population P systems in solving hard problems like NP-complete problems. Anyway, we expect efficient solutions to these problems to be found in the context of population P systems as well by using different combinations of rule types considered in this paper.

Acknowledgements

This research was supported by the Engineering and Physical Science Research Council (EPSRC) of United Kingdom, Grant GR/R84221/01, and by Molecular Computing Network (MolCoNet), European Union Contract IST-2001-32008.

References

[Alberts et al. 2002] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: "The Molecular Biology of The Cell. Fourth Edition"; Garland Publ. Inc., London (2002)

[Bernardini and Gheorghe 2004a] Bernardini, F., Gheorghe, M.: "Cell Communication in Tissue P systems and Cell Division in Population P Systems"; Second brainstorming week on membrane computing, Seville, 2-7 February 2004, Tech. Rep. 01/2004, Dept. Comp. Sci. & Art. Int., University of Seville (2004), 74–91

[Bernardini and Gheorghe 2004b] Bernardini, F., Gheorghe, M.: "Cell Communication in Tissue P systems: Universality Results"; Submitted (2004)

[Bernardini and Păun 2004] Bernardini, F., Păun, A.: "Universality of Minimal Symport/Antiport: Five Membranes Suffice"; Membrane Computing. International Workshop, WMC 2003, Tarragona, Spain, July 2003. Revised Papers, Lecture Notes in Computer Science 2933, Springer, Berlin (2004), 43–54

[Cavaliere 2002] Cavaliere, M.: "Evolution-Communication P Systems"; Membrane Computing. International Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, August 19-23, 2002. Revised Papers, Lecture Notes in Computer Science 2597, Springer, Berlin (2003), 134–145

[Freund et al. 2003] Freund, R., Kari, L., Oswald, M., Sosik, P.: "Computationally Universal P Systems without Priorities: Two Catalysts are Sufficient"; Submitted (2003)

[Frisco and Hoogeboom 2003] Frisco, P., Hoogeboom, J., H.: "Simulating Counter Automata by P Systems with Symport/Antiport"; Membrane Computing. International Workshop, WMC-CdeA 02, Curtea de Arges, Romania, August 19-23, 2002. Revised Papers, Lecture Notes in Computer Science 2597, Springer, Berlin (2003), 288–301

[Hopcroft and Ulmann 1979] Hopcroft, J., Ulmann, J.: "Introduction to Automata Theory, Languages, and Computation"; Addison-Wesley (1979)

[Martín-Vide et al. 2004] Martín-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): "Membrane Computing. International Workshop, WMC 2003, Tarragona, Spain, July 2003. Revised Papers"; Lecture Notes in Computer Science 2933, Springer, Berlin (2004)

[Păun 2000] Păun, Gh.: "Computing with Membranes"; Journal of Computer and System Sciences, 61, 1 (2000), 108–143

[Păun 2002] Păun, Gh.: "Membrane Computing. An Introduction"; Springer, Berlin (2002)

[Păun et al. 2004] Păun, Gh., Riscos-Núñez, A., Romero-Jiménez, A., Sancho-Caparrini, F. (eds.): "Second brainstorming week on membrane computing, Seville, 2-7 February 2004"; Technical Report 01/2004, Research Group on Natural Computing, University of Seville (2004) `http://www.gcn.us.es`

[Păun et al. 2003] Păun, Gh., Rozenberg, G., Salomaa, A., Zandron, C.(eds.): "Membrane Computing. International Workshop, WMC-CdeA 02, Curtea de Arges, Romania, August 19-23, 2002. Revised Papers"; Lecture Notes in Computer Science 2597, Springer, Berlin (2003)

[Rozenberg and Salomaa 1980] Rozenberg, G., Salomaa, A.: "The Mathematical Theory of L Systems"; Academic Press, New York (1980)

[Rozenberg and Salomaa 1997] Rozenberg, G., Salomaa, A. (eds.): "Handbook of Formal Languages"; 3 volumes, Springer, Berlin (1997)