Proceedings of Grammar Systems Week 2004, edited by E. Csuhaj-Varjú and Gy. Vaszil,
MTA SZTAKI, Budapest, pages 66 - 77, 2004.

Population P Systems and Grammar Systems

Francesco Bernardini, Marian Gheorghe

Department of Computer Science, The University of Sheffield
Regent Court, Portobello Street, Sheffield S1 4DP, UK
{f.bernardini,m.gheorghe}@dcs.shef.ac.uk

Abstract

The paper introduces two preliminary variants of population P systems with string-objects. The former one is inspired by CD grammar systems regarded as a population of interacting/cooperating cells that use rewriting rules to manipulate set of strings placed inside them. In this case, cells are restricted to communicate only by means of the environment and no direct communication among the cells can take place in the system. The second variant instead uses the notion of communication mediated by query symbols considered for PC grammar systems in order to define the features of bond making rules in context of population P systems with string-objects. Some preliminary results concerning the computational power of these population P systems are reported and some directions for future research are briefly discussed.

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 [10] and some fairly recent results can be found in [12], [8]. Membrane computing has been introduced with the aim of defining computing devices, called P systems, which abstract from the structure and the functioning of living cells [9]. A new variant of P systems, called population P systems, has been recently introduced in [1] that extends the existing notions of P systems in the following senses:

- the structure of the system is defined as an arbitrary graph rather than as a tree; each node in the graph corresponds in an one-to-one manner to a cell in the system; cells are the basic functional units of a population P systems and they are allowed to communicate alongside the edges of the graph;
- the graph defining the structure of the system can change during a computation; these changes involve both the set of nodes and the set of edges in the graph; new cells can therefore be introduced in the system and new links can be formed among these cells by altering in this way their communication capabilities.

From a biological point of view, population P systems can be considered as an abstraction for populations of of bio-units aggregated in more complex bio-entities.

Cells represent individuals in the population that are allowed to interact/cooperate according to a specific communication model. Links between the cells model the fact that, in order to communicate, these individuals need to "get in touch" somehow. Furthermore, the operations of cell differentiation, cell division and cell death are considered for population P systems as generic mechanisms to alter the nature and the number of individuals in the system [1].

Grammar systems is another active area of theoretical computer science that advocates the use of a grammatical approach to model distribution and cooperation in a computing system [3]. Grammar systems, in their basic variant called CD grammar systems, consist of a number of distinct grammars cooperating each other according to a given protocol in order to rewrite a common string shared by all the grammars in the system. The initial motivation for CD grammar systems were related to two-level grammars and to artificial intelligence issues. In this respect, CD grammar system can be seen as a model for describing autonomous agents cooperating each other in order to solve a common problem in such a ways that resembles the blackboard paradigm used in artificial intelligence [3]. Further variants of grammar system have been then proposed that introduce in the model extra features from other areas of parallel/distributed computing and/or with some biological inspiration [3], [4].

There are evident similarities between the P system model and the grammar system model: both of them define devices where computing resources are distributed among different individual components and interaction/cooperation between these components is fundamental to achieve successful and meaningful computations. In particular it is natural to think of a grammar systems as a population of interacting/cooperating grammars that work in a common shared environment. This is the approach we adopt in this paper where we investigate population P systems that use rewriting as the basic operation to manipulate sets of strings placed inside the cells in the system. All over the paper, the reader is supposed to be familiar with the basic notions of formal language theory, the notation commonly used in membrane computing and in grammar systems as well. We refer to [13], [6], [10], [3] for further details.

2 Modelling CD Grammar Systems

A CD grammar system [3] is a construct $\Gamma = (N, T, P_1, P_2, \ldots, P_n, S)$ where: N is a finite set of non-terminal symbols, T is a finite set of terminal symbols, P_i, for each $1 \leq i \leq n$, is a finite set of context-free rules of the form $X \to w$, with $X \in N$, $w \in (N \cup T)^*$, and $S \in N$ is the initial symbol of the system. Notice that, without loss of generality, CD grammar systems are presented here in the form that imposes the restriction of not rewriting the terminals of a component in the other ones. In the case of context-free rules, this restriction in fact does not alter the generative capacity of CD grammar systems (see Theorem 6.1 in [3]).

Each component in a CD grammar system Γ always operates according to a specific derivation mode $\delta \in \{k, \leq k, \geq k, *, t \mid k \geq 1\}$ adopted by the system Γ (see [3] for a formal definition of these derivation modes). Specifically, given two strings

$x, y \in (N \cup T)^*$ we write $x \Longrightarrow_{\Gamma}^{\delta} y$ if and only if $x \Longrightarrow_{P_i}^{\delta} y$, for some $1 \leq i \leq n$. Thus, the language $L_\delta(\Gamma)$ generated by the grammar system Γ using the derivation mode δ, is the language that contains all the strings $y \in T^*$ such that there exists a derivation in Γ of the form $S \Longrightarrow_{\Gamma}^{\delta} z_1 \Longrightarrow_{\Gamma}^{\delta} \ldots \Longrightarrow_{\Gamma}^{\delta} z_h \Longrightarrow_{\Gamma}^{\delta} y$, with $h \geq 0$, $z_j \in (N \cup T)^+$.

In this framework, a population P system model for systems of cooperating grammars can be defined in a straightforward manner by considering systems consisiting of a number cells with some finite sets of context-free rule and sharing a common environment; this environment is used to move the strings to be rewritten from one cell to another one and to collect the result of a computation. More precisely, an unstructured population P system is defined as a construct $\mathcal{P} = (V, T, E, C_1, C_2, \ldots, C_n)$ where: V a is finite set of symbols, $T \subseteq V$ is a finite set of terminal symbols, $E \subseteq V^*$ is a finite language assigned to the environment and, for each $1 \leq i \leq n$, $C_i = (R_i, F_i)$ with R_i a finite set of context-free rules of the form $a \to (v, out)$, for $a \in V$, $v \in V^*$, and F_i a finite set of filters of the form (a, in), for $a \in V$.

A computation in an unstructured population P system is performed by distributing to the cells, according to their respective sets of filters, the strings in the environment, where they are rewritten by means of some rules and immediately returned to the environment so that the process can be iterated. A filter $(a, in) \in F_i$, with $a \in V$, specifies that a string in the environment can enter cell i if and only if the symbol a is present inside that string; a rule $a \to (v, out)$ in R_i specifies that a string can be rewritten inside cell i by replacing the symbol a with the string v and, after that, the string has to exit cell i and be associated with the environment. The language generated by an unstructured population P system $\mathcal{P}$ is the language $L(\mathcal{P})$ that consists of all the strings in T^* that are produced inside the environment and that cannot enter the cells anymore.

Thus, by having this notion of population P systems, it is easy to verify that the following proposition holds.

Proposition 2.1. *Let Γ be a CD grammar systems with $n \geq 1$ components. It is always possible to construct an unstructured population P system $\mathcal{P}$ such that $L(\mathcal{P}) = L_1(\Gamma)$.*

On the other hand, it is not so immediate to simulate in the context of (population) P systems derivation modes other than the 1-mode. This might require the introduction of specific operations to manipulate the cells in the system or the introduction of specific control mechanisms to regulate the communication of strings from one place to another one. However, as we will see in the next section, population P systems are in general more powerful than CD grammar systems in terms of generative capacity.

3 Unstructured Population P Systems

In this section we introduce a more general notion of unstructured population P systems and we report some results concerning their computational power.

Definition 3.1. An unstructured population P system with string-objects is a construct

$$\mathcal{P} = (V, T, L, E, C_1, C_2, \ldots, C_n, R),$$

where:

1. V is a finite alphabet;
2. $T \subseteq V$ is a finite set of terminal symbols;
3. L is a finite set of labels that defines a set of possible types for the cells in the system;
4. $E \subseteq V^*$ is a finite language initially assigned to the environment;
5. $C_i = (M_i, t_i)$, for each $1 \leq i \leq n$, with $M_i \subseteq V^*$ a finite language defining the initial content of cell i, and $t_i \in L$ the initial type of cell i;
6. R is a finite set of rules of the forms:
   (a) $(\, a \to y \,)_t$, for $a \in V$, $v \in V^*$, $t \in L$ (*transformation rules* that allow a cell of type t to rewrite one of its internal strings by replacing the symbols a with the string v),
   (b) $(\, a \,)_t \to v\,(\,)_t$, for $a \in V$, $v \in V^*$, $t \in L$ (*output rules* that allow a cell of type t to rewrite one of its internal strings and move it into the environment),
   (c) $a(\,)_t \to (\, v \,)_t$, for $a \in V$, $v \in V^*$, $t \in L$ (*input rules* that allow a cell of type t to rewrite a string in the environment and add it to the content of the cell),
   (d) $(\, a \,)_t \to (\, v \,)_p$, for $a \in V$, $v \in V^*$, $t, p \in L$(*cell differentiation rules* that allow a cell of type t to rewrite one of its internal strings and change its type from t to p)
   (e) $(\, a \,)_t \to (\, v \,)_t\,(\, z \,)_t$, for $a \in V$, $v, z \in V^*$, $t \in L$ (*cell division rules* that allow cell of type t to rewrite one of its internal strings in order to produce two cells of type t, containing the same set of string as the originating one except for the string to be rewritten where the symbol a is replaced by the string v in one cell and by the string z in the other one).

This definition is obtained as a straightforward adaption to the case of string-objects of the definition given in [1] for population P systems with active cells in the case of symbols objects. In particular, communication of strings through the environment is achieved by means of a finite set of output and input rules: these rules allow a string to be rewritten inside a cell and then moved out from that cell or, alternatively, to be rewritten inside the environment and moved into a cell.

As usual, a computation in an unstructured population P system $\mathcal{P}$ is obtained by applying in a non-deterministic maximal parallel manner the rules in R to the strings in the system by starting from the initial configuration, with the further specification that each string is rewritten in a sequential way and that at most one rule of the

form (d), or (e) per each cell can be used at a time. The output of a computation is given by the set of strings in T^* that are produced inside the environment and that cannot enter the cells anymore; as usual in P systems with string-objects, we do not work with halting computations but we accept all the strings of the aforementioned form produced by any computation in $\mathcal{P}$. The language generated by $\mathcal{P}$ in this way is then denoted by $L(\mathcal{P})$. Then, we introduce the families of languages of the form $ELPP_{n,k}(op)$, with $n, k \geq 1$, and $op \subseteq \{a, b, c, d, e\}$ that represent the families of languages generated by (unstructured) population P systems where: the number of cells in a step of computation is always less than or equal to n, the cardinality of the set of possible types for the cells is at most k, and rules of the forms specified in op.

As well as this, we consider the families of languages of the form $ELP_n(tar)$ as the families of languages generated by standard rewriting P systems with at most n membranes where: the structure of the system is defined as a hierarchical arrangement of membranes represented as a tree, and communication of strings from one cell to another one is performed by using the target $here$, in_j, out. We refer to [10] for a formal definition of the basic model of rewriting P systems. Finally, we denote by MAT the family of languages generated by matrix grammars without appearance checking [6].

The first result we present here shows that unstructured population P systems without cell differentiation rules and cell division rules are no more than rewriting P systems.

Lemma 3.1. *$ELPP_{n,*}(\{a, b, c\}) \subseteq ELP_{n+1}(tar)$, for each $n \geq 1$.*

Proof. The proof is based on the observation that an unstructured population P system with $n \geq 1$ cells can be interpreted as being a rewriting P system with $n \geq 1$ elementary membranes embedded in a unique main membrane, which is the skin membrane of the P system and play the same role as the environment in the population P system. More precisely, given a population P system $\mathcal{P}$ with $n \geq 1$ cells $C_1, \ldots, C_n$ as specified in Definition 3.1, we construct a rewriting P system Π with $n + 1$ membranes labeled by $0, \ldots, n$ where:

- 0 is the label of the skin membrane;
- i, for each $1 \leq i \leq n$, is the label of an elementary membrane that is contained inside membrane 0;
- for each transformation rule $(a \to y)_i$ in R , with $1 \leq i \leq n$, there exists a corresponding rule $a \to (v, here)$ in R_i;
- for each output rule $(a)_i \to v()_i$ in R, with $1 \leq i \leq n$, there exists a corresponding rule $a \to (v, out)$ in R_i;
- for each input rule $a()_i \to (v)_i$ in R, with $1 \leq i \leq n$, there exists a rule $a \to (v, in_i)$ in R_0;

Notice that, without loss of generality, we are assuming $L = \{1, \ldots, n\}$ and each cell C_i to be of type i, for each $1 \leq i \leq n$. Moreover, in order to correctly simulate

the behaviour of $\mathcal{P}$, we add to R_0 a rule $a \to (a, out)$, for each symbol a in the system (i.e., at any moment a string produced inside the skin membrane can be sent out of the system), and we consider for the P system Π the set of terminal symbols $T' = \{ a \mid a \in T, \nexists a(\,)_t \to (\,a\,)_i \in R \}$, for T the set of terminal symbols of $\mathcal{P}$ (i.e., we accept only strings in the environment that are terminal and that cannot enter anymore the cells in the system). This means the language $L(\Pi)$ containing all the the strings in T'^* sent out of the system during all the possible computations in Π is exactly the language $L(\mathcal{P})$. □

Then, as a direct consequence of the equivalence $ELP_3(tar) = MAT$ established in the existing literature [7], [10] and the previous result, we obtain immediately the following result.

Corollary 3.1. $ELPP_{*,*}(\{a, b, c\}) \subseteq ELP_3(tar) = MAT$.

As well as this, we can prove the opposite inclusion: rewriting P systems can be simulated by unstructured population P systems without using cell differentiation and cell division.

Lemma 3.2. $ELP_n(tar) \subseteq ELPP_{n,n}(\{a, b, c\})$, *for each* $n \geq 1$.

Proof. Let Π be a rewriting P system with $n \geq 1$ membranes labeled in an one-to-one manner by $1, \ldots, n$ where 1 is the label of the skin membrane. We construct a population P system $\mathcal{P}$ with n cells where:

- $L = \{1, \ldots, n\}$;
- for each rule $a \to (v, here)$ in R_i, with $1 \leq i \leq n$, there exists a corresponding output rule $(\,a\,)_i \to v\$_{here_i}\,(\,)_i$ in R;
- for each rule $a \to (v, out)$ in R_i, with $1 < i \leq n$, there exists a corresponding output rule $(\,a\,)_i \to v\$_{out_i}\,(\,)_i$ in R;
- for each rule $a \to (v, out)$ in R_1, there exists a corresponding output rule $(\,a\,)_1 \to v\,(\,)_1$ in R;
- for each rule $a \to (v, in_j)$ in R_i, with $1 \leq i \leq n$, there exists a corresponding output rule $(\,a\,)_i \to v\$_{in_j}\,(\,)_i$ in R;
- there exists an input rule $\$_{here_i}\,(\,)_i \to (\,\lambda\,)_i$ in R, for each $1 \leq i \leq n$;
- there exists an input rule $\$_{in_i}\,(\,)_i \to (\,\lambda\,)_i$ in R, for each $1 \leq i \leq n$;
- there exists an input rule $\$_{out_j}\,(\,)_i \to (\,\lambda\,)_i$ in R, for each $1 \leq i \neq j \leq n$, with membrane i the membrane that contains membrane j.

The simulation of the P system Π by means of the population P system Π is done in the following way. We rewrite all the strings inside the cells in $\mathcal{P}$ in the same way as in Π by inserting in each of them a special symbol $\$_t$ carrying the target information t needed to move the strings in the right places. These strings are immediately moved out from the cells into the environment and, in the next step, they will be distributed

to the cells according to their respective target indications by using the input rules in R so that the process can be iterated. Moreover, the strings that are sent out of the skin membrane are moved into the environment without any special symbol $\$_t$, which means these strings will never be able to enter again the cells in the system. Therefore, the language $L(\mathcal{P})$ is exactly the language of terminal strings that are sent out of the system during all the possible computations in $\mathcal{P}$. □

Thus, by combining the previous result with Corollary 3.1, we obtain the following characterisation of the languages generated by unstructured population P systems without cell division rules and cell differentiation rules.

Corollary 3.2. $ELLP_{*,*}(\{a,b,c\}) = ELLP_{3,3}(\{a,b,c\}) = MAT.$

Finally, the case of population P systems with cell division rules and/or cell differentiation rules remains to be considered. In this respect, the sole operation of cell division is expected not to increase the power of population P systems, whereas population P systems with cell differentiation rules are expected to be computationally complete.

Conjecture 3.1. $ELLP_{*,*}(\{a,b,c,d\}) = RE.$

In fact, it should be easy to prove that population P systems with cell differentiation rules are able to simulate matrix grammars with appearance checking. This result, if proved, would be coherent with what was achieved in [1] where a similar universality result based on the operation of cell differentiation is provided for population P systems with symbol-objects.

4 Using Bond Making Rules

Direct communication among the cells (or membranes) in the system is one of the defining features of the membrane computing paradigm. Cells work in parallel on different sets (or multisets) of objects, which can then be moved from one cell to another one by means of some dedicated mechanisms typically expressed as a finite set of communication rules [10], [12], [8]. Moreover, population P systems [1] introduces the possibility of altering communication capabilities of the cells by modifying the set of links existing between the cells. This is done by considering a finite set of bond making rules which are used after each application of transformation/communication rules to the objects contained in the cells; this makes possible to modify the set of edges in the graph defining the structure of the system. Notice that a communication rule inside a cell can be used if and only if the cell is linked to some other cells by means of some edges in the underlying graph.

In grammar systems, these features of parallelism and communication among the components were firstly considered by introducing the notion of parallel communicating grammar systems (PC grammar systems, for short): a model for networks of Chomsky grammars communicating strings by emerging requests [3]. More precisely, in each step, each grammar in the system rewrites its string and communication is done by requests through so-called query symbols, each one of them referring to a

specific grammar in the system. When a query symbol appears in the string of a grammar, the rewriting process stops and one or more communication steps are performed by replacing all occurrences of the query symbols with the current string of the queried grammars providing that this string does not contain any query symbol. When no more query symbols are present in the system the rewriting process can start again.

From a P system point of view, this communication by request can be considered as a mechanism to open communication channels between the cells in the system. Specifically, here we want to use query symbols for defining bond making rules that make possible to link two cells provided that these cells contain a pair of corresponding query symbols. Then, once a link has been established, communication between the two cells can take place by rewriting their respective query symbols by means of some particular communication rules associated with the cells.

Definition 4.1. A population P system with bond making rules is a construct

$$\mathcal{P} = (V, T, Q, \alpha, C_1, C_2, \ldots, C_n)$$

where:

1. V is a finite alphabet;
2. $T \subseteq V$ is a finite set of terminal symbols;
3. Q is a finite of set of query symbols;
4. α is a finite set of bond making rules of the form $(i, q_1; q_2, j)$, with $1 \leq i \neq j \leq n$ and $q_1, q_2 \in Q$;
5. $C_i = (M_i, R_i, S_i)$, for each $1 \leq i \leq n$, with:
   (a) $M_i \subseteq V^*$ a finite language defining the initial content of cell i;
   (b) R_i a finite set of transformation rules of the forms $a \to v$, $a \to v\,q$, $a \to (v, out)$, for $a \in V$, $v \in V^*$, and $q \in Q$;
   (c) S_i is a finite set of communication rules of the forms $q \to v$, $q \to v\,q'$, for $q, q' \in Q$, and $v \in V^*$.

Here, with respect to Definition 3.1, we are considering population P systems without cell differentiation and cell division where the number and the type of cells in the system cannot vary during a computation. As well as this, we do not need a notion of environment as cells are allowed to communicate directly by using their respective sets of communication rules in combination with the set of bond making rules. In this respect, the structure of a population P system with bond making rules $\mathcal{P}$ is given by the the set of communication links existing among the cells in the system that are created time by time by applying the bond making rules in α.

Each cell C_i, with $1 \leq i \leq n$, gets assigned a finite language M_i that defines its initial content. Each string contained in cell i is rewritten by using the transformation rules in R_i. A rule $a \to v \in R_i$ specifies that a string containing a symbol a can be rewritten inside cell i by replacing the symbol a with the string v. In a

similar way, a rule $a \to (v, out) \in R_i$ allows cell i to rewrite one of its internal strings but, after that, the string has to exit the cell without any chance to enter again any cell of the system. Finally, a rule $a \to v\,q \in R_i$ makes possible to rewrite a string inside cell i and introduce in this string the query symbol q. After the application of one of these rules, the rewriting of the string containing the query symbol q inside cell i stops and it will restart only when the symbol q has been satisfied and removed from the string. However, this does not stop the rewriting of the other strings contained in cell i. In order to satisfy a query symbol, the system operates as follows. Consider a bond making rule $(i.q_1; q_2, j)$, with $q_1, q_2 \in Q$, $1 \leq i \neq j \leq n$, and suppose that, after one application of the respective transformation rules in R_i and in R_j, a string $x\,q_1\,y$ is produced inside cell i and a string $u\,q_2\,z$ is produced inside cell j, with $x, y, u, z \in V^*$. This means a bond between cell i and cell j can be created by connecting the string $x\,q_1\,y$ with the string $u\,q_2\,z$, and the query symbol q_1 can be satisfied by using the communication rules in S_j whereas the symbol q_2 can be satisfied by using the rules in S_i. Notice that, for each $1 \leq k \leq n$, the set S_k contains both rules of the form $q \to v$ that remove the query symbol from the string, and rules of the form $q \to v\,q'$ that introduce a new query symbol, which makes the string to wa!it for another communication to take place.

As usual, we assume a non-deterministic maximal parallel strategy for the application of the rules:

- in each step, in each cell i, with $1 \leq i \leq n$, each string that can be rewritten by some rules in R_i must be rewritten by using one rule in R_i non-deterministically chosen (i.e., each string inside cell i is rewritten in a sequential manner); at the same time, each query symbol in each cell j linked to cell i that can be satisfied must be satisfied by using one communication rule in S_i non-deterministically chosen;

- after each application of the transformation/communication rules, for each bond making rule $(i.q_1; q_2, j)$, with $q_1, q_2 \in Q$, $1 \leq i \neq j \leq n$, we establish a link between cell i and cell j for each pair of strings $x\,q_1\,y$, $u\,q_2\,z$, with the former one contained in cell i and the latter one contained in cell j, for $x, y, u, z \in V^*$; the restriction here is that the same string containing a query symbol can be used by only one bond making rule non-deterministically chosen.

Therefore, a step of computation in a population P system with bond making rules is done in two separate stages: a stage where transformation/communication rules are applied to the strings contained inside the cells and a stage where bond making rules are used in order to create links between the cells in the system. Notice that the query symbols introduced by the application of the transformation rules can be satisfied only in the next step of computation after having used the bond making rules. Furthermore, these bond making rules make possible to create multi-links between the cells that are represented by sets of pairs of corresponding strings.

The language generated by a population P system with bond making rules $\mathcal{P}$ is the language $L(\mathcal{P})$ containing all the strings sent out of the cells during all the possible computations in $\mathcal{P}$. The family of languages generated by population P systems with bond making rules and with at most $n \geq 1$ cells is denoted by $ELPP_n(query)$.

Next, we prove that these population P systems are quite powerful from a computational point of view: systems with only two membranes can generate the whole family of languages in MAT.

Lemma 4.1. $MAT \subseteq ELPP_2(query)$.

Proof. Consider, without loss of generality, a matrix grammar without appearance checking G in binary normal form. This is because it is known from [6] that, for each matrix grammar generating a language L, there always exists an equivalent matrix grammar in binary normal formal generating the same language L. Moreover, we assume all the matrices in G but the initial matrix $(S \to XA)$ to be labeled in an one-to-one manner by values in $\{1, 2, \ldots, m\}$. We construct a population P system $\mathcal{P}$ with 2 cells where:

- $C_1 = (M_1, R_1, S_1)$ with
  - $M_1 = \{X\}$, for $(S \to XA)$ the initial matrix of G;
  - $R_1 = \{\, X \to X_i \mid X \in N_1, 1 \leq i \leq n \,\}$;
  - $S_1 = \{\, A_i \to v \mid i : (X \to Y, A \to v)$ is a matrix in $G, 1 \leq i \leq n \,\}$
    $\cup \{\, A_i \to v\, f \mid i : (X \to \lambda, A \to v)$ is a matrix in $G, 1 \leq i \leq n \,\}$;
- $C_2 = (M_2, R_2, S_2)$ with
  - $M_2 = \{A\}$, for $(S \to XA)$ the initial matrix of G;
  - $R_2 = \{\, A \to A_i \mid A \in N_2, 1 \leq i \leq n \,\} \cup \{\, f \to (f, out) \,\}$;
  - $S_2 = \{\, X_i \to Y \mid i : (X \to Y, A \to v)$ is a matrix in $G, 1 \leq i \leq n \,\}$
    $\cup \{\, X_i \to \lambda \mid i : (X \to \lambda, A \to v)$ is a matrix in $G, 1 \leq i \leq n \,\}$;

for $Q = \{\, X_i, A_i \mid X \in N_1, A \in N_2, 1 \leq i \leq n \,\}$. Now, by having this construction, it is easy to see that the population P system $\mathcal{P}$ correctly simulates the matrix grammar G and therefore we have $L(\mathcal{P}) = L(G)$. □

At the moment, we are not able to provide an upper bound for the generative capacity of population P systems with bond making rules and answer the question whether they are more than usual rewriting P systems or not.

5 Final Remarks

Membrane computing and grammar systems are two active areas of theoretical computer science, with different starting points, but with several similarities (both areas deal with distributed computing devices, where such notions as parallelism, cooperation, decentralisation are crucial). Nevertheless, as we have seen, important differences between the two models emerge from a deeper investigation especially in terms of computational power. Population P systems even of a very basic form are able to characterise the whole family of languages generated by matrix grammars without appearance checking (see Corollary 3.2). A similar result was already established for the basic model of rewriting P systems as reported in Corollary 3.1. The power

of (population) P systems is in fact determined by the underlying communication structure that allows the cells to exchange strings in a controlled way. In this sense, P systems with string-objects appears to be more similar to the already existing notions of networks of evolutionary processors [5], [2].

On the other hand, a general claim in the area of P systems with string-objects is that rewriting is not enough to achieve the computational completeness but it is necessary to introduce in rewriting P systems some extra features such as priority, replicated rewriting or conditional communication [10]. In the case of unstructured population P systems, we expect the universality to be obtained by including the operation of cell differentiation, which makes possible to change the types of the cells in the system (see Conjecture 3.1). Furthermore, the claim "rewriting is not enough" might represent a motivation for introducing grammar system features in (population) P systems. In this respect, Section 4 introduces a variant of population P systems where the notion of query symbols considered for PC grammar systems is used to define bond making rules and the related communication model. Notice that, according to Definition 4.1, communication in a population P system with bond making rules consists in just rewriting a query symbol in a cell by means of some rules associated with another cell without an effective movement of the strings from a cell to another one. In other words, a string containing a query symbol is temporarily given access to the rules in a cell different from the one where the string is placed. Nevertheless, we could easily define a variant of this model where strings are moved from a cell to another one by replacing a query symbol in a cell with all the strings contained in the queried cell and producing all the resulting strings inside the querying cell. As well as this, we might consider multisets of strings instead of sets of strings as, for instance, proposed in [11] where PC grammar systems with multisets of strings are investigated in relationship with P systems.

Finally, further investigations of population P systems with string-objects might be directed to clarify the role of the environment in achieving successful and meaningful computation; this might be done by pointing out analogies with the existing models of eco-grammar systems [4].

Acknowledgements

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

References

[1] Bernardini, F., Gheorghe, M., (2004). Population P systems. *Journal of Universal Computer Science*, **10**, 509-539.

[2] Castellanos, J., Martin-Vide, C., Mitrana, V., Sempere, J., M., (2003). Networks of Evolutionary Processors. *Acta Informatica*, **39**, 517-529.

[3] Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, Gh., (1994). *Grammar Systems. A Grammatical Approach to Distribution and Cooperation.* Gordon and Breach, London.

[4] Csuhaj-Varjú, E., Kelemen, J., Kelemenova, A., Păun, Gh., (1997). Eco-Grammar Systems: A Grammatical Framework for Studying Life-Like Interactions. *Artificial Life*, **3**, 1-28.

[5] Csuhaj-Varjú, E., Salomaa, A., (1997). Networks of Parallel Languages Processors. In *New Trends in Formal Languages* (Pň, Gh., Salomaa, A., eds.), Lecture Notes in Computer Science, **1218**, Springer, Berlin, Heidelberg, New York, 299-318.

[6] Dassow, J., Păun, Gh., (1989). *Regulated Rewriting in Formal Language Theory.* EATCS Monograph in Theoretical Computer Science, Springer-Verlag, Berlin, Heidelberg, New York.

[7] Madhu, M., (2003). *Studies of P Systems as a Model of Cellular Computing*, PhD Thesis, Indian Institute of Technology, Madras, India.

[8] Martin-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G., Salomaa, A., eds., (2004). *Membrane computing. International workshop, WMC 2003, Tarragona, Spain, July 2003. Revised papers.* Lecture Notes in Computer Science, **2933**, Springer, Berlin, Heidelberg, New York.

[9] Păun, Gh., (2000). Computing with Membranes. *Journal of Computer and System Sciences*, **61**, 108–143.

[10] Păun, Gh. (2002). *Membrane Computing. An Introduction.* Natural Computing Series, Springer, Berlin, Heidelberg, New York.

[11] Păun, Gh., (2004). Grammar Systems vs. Membrane Computing: A Preliminary Approach. In *Pre-Proceedings of Grammar Systems Week 2004, Budapest, Hungary, July 5-9, 2004* (Csuhaj-Varjú, E., Vaszil, G., eds.), MTA SZTAKI, Budapest, 225-244.

[12] Păun, Gh., Rozenberg, G., Salomaa, A., Zandron, C., eds., (2003). *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, Heidelberg, New York.

[13] Rozenberg, G., Salomaa, A., eds., (1997). *Handbook of Formal Languages.* 3 volumes, Springer, Berlin, Heidelberg, New York.