KYBERNETIKA — VOLUME *48* (2012), NUMBER 3, PAGES 502–517

POSITIONED AGENTS IN ECO-GRAMMAR SYSTEMS WITH BORDER MARKERS AND PURE REGULATED GRAMMARS

Miroslav Langer, Alica Kelemenová

In this paper we follow our previous research in the field of positioned agents in the eco-grammar systems and pure grammars. We extend model of the positioned eco-grammar systems by boundary markers and we introduce bordered positioned eco-grammar systems (BPEG systems, for short) and that way we show one of the possible answers to the question stated in [9]. Namely we compare generative power of the BPEG systems with three types of pure regulated grammars with appearance checking.

Keywords: positioned eco-grammar systems, bordered positioned eco-grammar systems, pure regulated grammars with appearance checking

Classification: 22E46, 53C35, 57S20

1. INTRODUCTION

Positioned eco-grammar systems (PEG systems, for short) were introduced in [10]. Similarly as in eco-grammar systems, the motivation is an attempt to describe interplay between evolving environment and community of agents living in this environment, whereas we focus on agent's position in the environment. We combine the approaches from PM-colonies (see [12]) and eco-grammar systems (see [2, 3]). The environment of the PEG system is represented by 0L scheme and position of each agent in the environment is given by its identifier.

In this paper we follow our previous results concerning PEG systems and their relation to pure (regulated) grammars. In [9] we showed that the families of languages generated by the pure (regulated) grammars without appearance checking are proper subsets of PEG languages. In the conclusion we have stated an open question; the relation between the language classes of the PEG systems and pure regulated grammar with appearance checking. Motivated by PM colonies which work with the border markers, in the present paper we study slightly modified positioned eco-grammar systems with additional border markers. This allows us more thoroughly to search the environment and it gives possibility to present relation to languages of pure regulated grammars with appearance checking.

We introduce bordered positioned eco-grammar systems (BPEG systems, for short) and we compare generative capacity of the BPEG systems with the generative capacity

of pure regulated grammars with appearance checking. We show that the families of languages generated by pure regulated grammars with appearance checking are (proper) subsets of the family of BPEG languages.

2. PRELIMINARIES ON PURE GRAMMARS

We assume that the reader is familiar with formal language theory including Lindenmayer systems (see [7, 13]) and regulated grammars (see [4, 6]).

In the present section we recall the notion of the pure grammars with regulated rewritings, namely pure matrix grammars with appearance checking, the pure programmed grammars with appearance checking and the pure random context grammars with appearance checking. We summarize results on the generative power of these grammars.

A pure grammar is a grammar with only one alphabet, a finite set of axioms and with the sequential derivation. All derived words belong to the language generated by the pure grammar. To specify different types of the pure grammars we start with the pure context-free grammars.

A *pure context-free grammar* is a triple $G = (V, P, S)$, where V is a finite nonempty alphabet, S is a finite set of words from V^* called axioms and P is a finite set of context-free productions of the form $a \to w,\ a \in V,\ w \in V^*$.

We say that $x \in V^+$ directly derives $y \in V^*$, denoted as $x \Longrightarrow y$, if $x = z_1 a z_2$, $y = z_1 w z_2$ and $a \to w \in P$ for some $z_1, z_2, w \in V^*$ and $a \in V$. The relation $\Longrightarrow^*$ denotes the reflexive and transitive closure of the relation $\Longrightarrow$. The language of the pure context-free grammar G is $L(G) = \{y : x \Longrightarrow^* y,\ x \in S\}$.

Typical example of the pure context-free language is the language $\{a^i c b^i : i \geq 0\}$. On the other hand, the language $\{a^i b^i : i \geq 0\}$ is not included in the pure context-free language class.

To present pure grammars with regulated rewriting, we start with the matrix (context-free) grammars, where the order of the application of productions in the derivation is specified by finite sequences of productions. All the productions in the chosen sequence have to be used in a given order in the derivation with only exception of the productions from the given set. These productions can be omitted in the case they are not applicable in an actual string. Formally:

A *pure matrix grammar* is a quadruple $G = (V, M, S, F)$, where V is a finite nonempty alphabet, S is a finite set of axioms over V, $M = \{m_1, \ldots, m_s\}$ is a finite set of finite sequences of productions $m_i, 1 \leq i \leq s$, of the type: $(a_{i,1} \to w_{i,1}, a_{i,2} \to w_{i,2}, \ldots a_{i,k_i} \to w_{i,k_i})$ for $a_{i,j} \in V, w_{i,j} \in V^*, 1 \leq j \leq k_i$ and F is a subset of occurrences of in $m_i,\ 1 \leq i \leq s$ from M.

We say that $x \Longrightarrow_{m_i} y$ for $1 \leq i \leq s$ iff $x = y_0 \Longrightarrow y_1 \Longrightarrow y_2 \ldots \Longrightarrow y_{k_i} = y$, where either $y_{j-1} = z_j a_{i,j} z'_j,\ y_j = z_j w_{i,j} z'_j$ for $a_{i,j} \to w_{i,j}$ or $a_{i,j}$ does not occur in y_{j-1}, $y_j = y_{j-1}$ and $a_{i,j} \to w_{i,j} \in F$ for $1 \leq j \leq k_i$.

The language of pure matrix grammar is the set of all words y which are obtained by iterative application of matrices to the words from S and of all intermediate words y_j of these applications.

The set F has an appearance checking function for the pure matrix grammar.

The class of the all pure matrix context-free languages with the appearance checking is denoted by $\mathcal{L}(pM, CF, ac)$.

The class of the all pure matrix context-free languages without the appearance checking is denoted by $\mathcal{L}(pM, CF)$.

Furthermore we present pure programmed grammars, where each production is associated with set of productions which can be applied in the forthcoming derivation step and the other set of productions to be applied in the case that the production itself cannot be applied to the current word.

A *pure programmed grammar* is a triple $G = (V, P, S)$, where V is a finite nonempty alphabet, S is a finite set of axioms over V, P is a finite set of productions of the form $(r : a \to w, \sigma(r), \varphi(r))$, where r is the unique label of this production, $a \in V$, $w \in V^*, \sigma(r)$ and $\varphi(r)$ are subsets of the set of labels of the productions in P.

The language defined by the pure programmed grammar is the set of all words y such that there exists a derivation $s = y_0 \Longrightarrow_{r_1} y_1 \Longrightarrow_{r_2} \ldots \Longrightarrow_{r_n} y_n = y$ where $s \in S$, $(r_i : a_i \to w_i, \sigma(r_i), \varphi(r_i)) \in P$ for $1 \leq i \leq n$ and

$y_{i-1} = z_1 a_i z_2$, $y_i = z_1 w_i z_2$, $r_{i+1} \in \sigma(r_i)$, or

a_i does not occur in y_{i-1}, $y_i = y_{i-1}$ and $r_{i+1} \in \varphi(r_i)$, for $1 \leq i \leq n$.

The set $\varphi(r)$ has an appearance checking function for the pure programmed grammar.

The class of the all pure programmed context-free languages with appearance checking is denoted by $\mathcal{L}(pP, CF, ac)$.

The class of the all pure programmed context-free languages without appearance checking is denoted by $\mathcal{L}(pP, CF)$.

Last type of grammars with regulated rewriting considered in the present paper is the pure random context context-free grammars.

A *pure random context grammar* is a triple $G = (V, P, S)$, where V is a finite nonempty alphabet, S is a finite set of axioms, P is a finite set of productions of the form $(a \to w, Q, R)$, where $a \in V$, $w \in V^*, Q \subseteq V$ and $R \subseteq V$.

We say that x directly derives y, $x \Longrightarrow_{rc} y$ for $x, y \in V^*$ if there is $(a \to w, Q, R)$ in P such that $x = x'ax'', y = x'wx''$, every letter of Q occurs in $x'x''$ and no letter of R occurs in $x'x''$.

The language generated by the pure random context context-free grammar G is defined as $L(G) = \{y : x \Longrightarrow^*_{rc} y, x \in S\}$, where $\Longrightarrow^*_{rc}$ is a reflexive and transitive closure of relation $\Longrightarrow_{rc}$.

The set R has an appearance checking function for the pure random context grammars.

The class of all pure random context context-free languages with appearance checking is denoted by $\mathcal{L}(pRC, CF, ac)$.

The class of all pure random context context-free languages without appearance checking is denoted by $\mathcal{L}(pRC, CF)$.

Proposition 2.1. (Dassow and Păun [4]) $\mathcal{L}(pCF) \subset \mathcal{L}(pM, CF, ac) \cap \mathcal{L}(pP, CF, ac) \cap \mathcal{L}(pRC, CF, ac)$

Classes of languages $\mathcal{L}(pM, CF, ac), \mathcal{L}(pP, CF, ac), \mathcal{L}(pRC, CF, ac)$ are pairwise not comparable.

The following properties hold (see [4]):

- $L_1 = \{a^n b^n c^n : n \geq 1\} \cup \{a^{n+1} b^n c^n : n \geq 1\} \cup \{a^{n+1} b^{n+1} c^n : n \geq 1\}$ belongs to $\mathcal{L}(pM, CF) - (\mathcal{L}(pP, CF, ac) \cup \mathcal{L}(pRC, CF, ac))$

- $L_2 = \{a, a^5\} \cup \{a^{7+10n} : n \geq 0\} \cup \{a^{11+10n} : n \geq 0\}$ belongs to $\mathcal{L}(pP, CF) - (\mathcal{L}(pM, CF, ac) \cup \mathcal{L}(pRC, CF, ac))$
- $L_3 = \{bba\} \cup \{a^{4+3n}ba^{1+3m} : n \geq 0, m \geq 0\} \cup \{ba^{5+3k} : k \geq 0\}$ belongs to $\mathcal{L}(pRC, CF) - (\mathcal{L}(pM, CF, ac) \cup \mathcal{L}(pP, CF, ac))$.

For the proofs and further results on pure grammars we refer to [4, 6].

In the next section we will use also generative systems (grammars) with totally parallel rewriting steps, known as L systems (see [7, 13]).

0L system, L system with no interaction, is a construct $G = (V, P, w_0)$, where V is a finite nonempty alphabet, w_0 is a word over V and P is a finite set of rules of the form $a \to w$, where $a \in V$ and $w \in V^*$. We say that x directly derives y in the 0L system, $x \Longrightarrow y$ for $x, y \in V^*$ if $x = x_1x_2 \dots x_n,\ y = w_1w_2 \dots w_n$ and $x_i \to w_i \in P$ for $1 \leq i \leq n$.

3. BORDERED POSITIONED ECO-GRAMMAR SYSTEMS

In the present section we will deal with the bordered positioned eco-grammar systems. They are modification of the earlier introduced positioned eco-grammar systems ([10]), motivated by endmarkers of PM colonies (see [12]). Further information on eco-grammar systems can be found also e. g. in [2, 3].

Definition 3.1. Bordered positioned eco-grammar system ($BPEG$ system, for short) of degree m, $m \geq 1$, is an $(m+4)$-tuple $\Sigma = (V_E, \#, N_B, E, B_1, \dots, B_m)$, where

- $V_E \cup \{\#\}$ is a finite nonempty alphabet of the environment,
- $\#$ is the special border marker,
- $N_B = \{[j] : 1 \leq j \leq m\}$ is the set of identifiers of agents, $[j]$ defines position of the jth type agent in the environment,
- $E = (V_E \cup \{\#\}, P_E)$ is a $0L$, where the rule $\# \to \#$ is the only rule for the border marker used in P_E,
- $B_j = ([j], Q_j)$, is the jth type agent for $1 \leq j \leq m$ and Q_j is a set of rules of the form:
  - $a[j]b \to u$, where $ab \in V_E$ is a symbol marking (left or right) vicinity with the agent $[j]$ and $u \in (V_E \cup N_B)^*$, or
  - $\#[j] \to \#u$ or $[j]\# \to u\#$, where $u \in (V_E \cup N_B)^*$, the rules for the interaction with border marker.

Note that the BPEG system requires at least one agent by the definition.

A configuration of the BPEG system is introduced as follows:

Definition 3.2. A configuration of the bordered positioned eco-grammar system $\Sigma = (V_E, \#, N_B, E, B_1, \dots, B_m)$ is a string $\#v\#$, where $v \in (V_E \cup N_B)^*$. The starting configuration is called an axiom.

Agents in the BPEG system will work parallel. Each agent appearing in an actual string has to rewrite one symbol on its right or left context in each derivation step, otherwise the derivation is blocked.

We describe the derivation step in $BPEG$ system as follows:

Definition 3.3. A derivation step of the bordered positioned eco-grammar system $\Sigma = (V_E, \#, N_B, E, B_1, \ldots, B_m)$ is a binary relation $\Longrightarrow_\Sigma$ on $\#(V_E \cup N_B)^*\#$, such that $\#w\# \Longrightarrow_\Sigma \#w'\#$ iff

- $w = \alpha_0 a_1[j_1]b_1\alpha_1 \ldots \alpha_{n-1}a_n[j_n]b_n\alpha_n$, where
  $\alpha_k \in V_E^*$ for $0 \leq k \leq n$ and $a_k b_k \in V_E, [j_k] \in N_B, 1 \leq k \leq n$,
- $w' = \alpha_0'\beta_1\alpha_1' \ldots \alpha_{n-1}'\beta_n\alpha_n'$, where
  $a_k[j_k]b_k \to \beta_k \in Q_k$ for $1 \leq k \leq n$ and $\alpha_k \Rightarrow_E \alpha_k'$ for $0 \leq k \leq n$.

A rule $\#[j_1] \to \#\beta_1 \in Q_1$ is used for $\alpha_0 a_1 b_1 = \varepsilon$ and $[j_n]\# \to \beta_n\# \in Q_n$ is used for $a_m b_m \alpha_m = \varepsilon$.

By $\Longrightarrow^*$ we denote the reflexive and transitive closure of the relation $\Longrightarrow$.

All agents work in parallel. Each agent rewrites one symbol on its right or left hand side together with its own identifier, in each derivation step. The rest of the symbols (i. e. those not touched by agents) are rewritten by the rules of the environment.

If two agents want to rewrite the same symbol the derivation is blocked as well as in the case when there is no rule for the agent in its actual context.

The language defined by a bordered positioned eco-grammar system is given by all words produced by the system from the axiom, ignoring agents identifiers and border markers.

Definition 3.4. The language defined by the bordered positioned eco-grammar system $\Sigma = (V_E, \#, N_B, E, B_1, \ldots, B_m)$ and the axiom $\#w\#, w \in (V_E \cup N_B)^*$ is a set of strings:

$$L(\Sigma, \#w\#) = \{\gamma(u) : u \in (V_E \cup N_B)^*, \#w\# \Rightarrow_\Sigma^* \#u\#\},$$

where γ is the morphism such that $\gamma(a) = a$ for $a \in V_E$ and $\gamma(b) = \varepsilon$ for $b \in N_B$.

The family of languages defined by bordered positioned eco-grammar systems ($BPEG$ languages) is denoted $\mathcal{L}(BPEG)$.

Even though the relation between positioned eco-grammar systems and bordered positioned eco-grammar systems was not investigated, it is obvious that $\mathcal{L}(PEG) \subseteq \mathcal{L}(BPEG)$.

4. BPEG SYSTEMS VERSUS PURE REGULATED GRAMMARS WITH APPEARANCE CHECKING

Strings produced in all derivation steps of BPEG systems are considered in BPEG languages. From that point of view, to compare them with adequate sequential devices, we consider regulated derivatives of pure context-free grammars of Chomsky hierarchy.

As we already mentioned in Section 2, Proposition 2.1, each pair of families of the following pure context-free languages (each one with appearance checking): matrix, random context, programmed are incomparable. In the present section we prove that all these pure regulated context-free language families with appearance checking are the subsets of the family of BPEG languages.

Lemma 4.1. The family of languages generated by the pure programmed context-free grammars with appearance checking is a subset of the family of languages generated by the bordered positioned eco-grammar systems.
$\mathcal{L}(pP, CF, ac) \subseteq \mathcal{L}(BPEG)$.

P r o o f. Consider $G = (V, P, S)$ a pure programmed context-free grammar with appearance checking, where P consists of the finite set of productions of the form $(r : a \to w, \sigma(r), \varphi(r)), 1 \leq r \leq n$.

We construct a BPEG system

$$\Sigma = (V, \#, N_B, E, B_I, B_D, B_1, \ldots, B_n, B_{1_R}, \ldots, B_{n_R}, B_{1_C}, \ldots, B_{n_C}),$$

with $3n + 2$ agents, where $n = |P|$ is number of production of grammar G and we determine the axiom $\#\alpha\#$, such that $L(\Sigma, \#\alpha\#) = L(G)$. The alphabet V of the BPEG system Σ is identical with that one of the pure grammar G. The non-evolving environment of BPEG system Σ is determined by the 0L scheme $E = (V \cup \{\#\}, \{a \to a : a \in (V \cup \{\#\})\})$.

The BPEG system Σ has five types of agents: the initiating agent B_I, the deleting agent B_D, and the returning agent B_{i_R}, the checking agent B_{i_C} and the simulating agent B_i for each production of the grammar G. So the corresponding agents positions in Σ form the set $N_B = \{[D], [I], [1], \ldots, [n], [1_R], \ldots, [n_R], [1_C], \ldots, [n_C]\}$.

The agents are specified as follows:

- *Initiating agent* $B_I = ([I], Q_I)$, where
  $Q_I = \{[I]a \to [i_R]y : a \in V, y \in S, 1 \leq i \leq n\}$.

- *Deleting agent* $B_D = ([D], Q_D)$, where $Q_D = \{[D]a \to \varepsilon : a \in V\}$.

For production $(i : a_i \to w_i, \sigma(i); \varphi(i)) \in P, 1 \leq i \leq n$, we define:

- *Returning agent* $B_{i_R} = ([i_R], Q_{i_R})$, where
  $Q_{i_R} = \{b[i_R] \to [i_R]b : b \in V\} \cup \{\#[i_R] \to \#[i_C]\}$
  It moves to the left # and then transforms to the checking agent.

- *Checking agent* $B_{i_C} = ([i_C], Q_{i_C})$, where
  $Q_{iC} = \{[i_C]b \to b[i_C] : b \in (V \setminus \{a_i\})\} \cup \{[i_C]a_i \to [i]a_i\} \cup \{[i_C]\# \to [j_R]\# : j \in \varphi = (i)\}$
  It moves to the right and it transforms to the simulating agent immediately after finding a_i on its right neighbour or to the returning agent of the next used production, otherwise.

- *Simulating agent* $B_i = ([i], Q_i)$, where
  $Q_i = \{[i]a_i \to [r_R]w_i : r \in \sigma(i)\} \cup \{[i]b \to b[i] : b \in V\} \cup \{c[i] \to [i]c : c \in V\}$
  It replaces one of the occurrences of a_i to w_i and at the same derivation step it transforms to the returning agent corresponding to the rule which will be used in the next derivation step in G.

The axiom is $\#\alpha\# = \#[I]a_1[D]a_2 \dots [D]a_k\#$, where $a_1a_2 \dots a_k \in S$ is some of the axioms of the grammar G.

We prove the equality $L(G) = L(\Sigma, \#\alpha\#)$ by showing both set inclusions. First we call attention to the fact that by the construction of Σ, its environment is stable and all changes in the derivation are done by the agents.

The inclusion $L(G) \subseteq L(\Sigma, \#\alpha\#)$: Assume that $w \in L(G)$ and its derivation in G is $y = w_0 \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w_s = w$. We will find corresponding derivation of w in Σ. To get y in Σ from the axiom $\#\alpha\#$ we simply use initiating and deleting agents in the following derivation step

$$\#\alpha\# = \#[I]a_1[D]a_2 \dots [D]a_k\# \Rightarrow \#[i_R]y\#,$$

where the agent $[i_R]$ is chosen so that the number i identifies the rule $(i : a_i \to w_i, \sigma(i), \varphi(i))$ used in the step $y \Rightarrow w_1$ in G.

Now it is sufficient to describe the derivation $\#[j_R]w_i\# \Rightarrow^* \#[t_R]w_{i+1}\#$ in Σ which corresponds to the derivation step $w_i \Rightarrow w_{i+1}$ and uses the production $(j : a \to w, \sigma(j), \varphi(j))$, $0 \le i \le s-1$. The production used in the $(i+1)$-st step of G is t. For $w_i = u_iav_i$ (a is a letter of w_i) and $t \in \sigma(j)$ we have in Σ derivation:

$$\#[j_R]w_i\# \Rightarrow \#[j_C]w_i\# \Rightarrow^* \#u_i[j_C]au_i\# \Rightarrow \#u_i[j]au_i\# \Rightarrow^* u_i'[j]au_i' \Rightarrow u_i'[t_R]wu_i' \Rightarrow^* \#[t_R]w_{i+1}\#.$$

For string w_i which does not contain a and $t \in \varphi(j)$ we have $w_{i+1} = w_i$ and corresponding derivation in Σ.

$$\#[j_R]w_i\# \Rightarrow \#[j_C]w_i\# \Rightarrow^* \#w_i[j_C]\# \Rightarrow \#w_i[t_R]\# \Rightarrow^* \#[t_R]w_i\# = \#[t_R]w_{i+1}\#.$$

In these two last derivations we obtain word w_i or w_{i+1} by eliminating the agents position symbols in derived words. As we have shown we are able to simulate correctly all derivation steps of G in Σ and thus we can prove the inclusion $L(G) \subseteq L(\Sigma, \#\alpha\#)$.

To prove the opposite inclusion $L(\Sigma, \#\alpha\#) \subseteq L(G)$ we assume that $w \in L(\Sigma, \#\alpha\#)$ and its derivation in Σ is of the form

$$\#\alpha\# = \#[I]a_1[D]a_2 \dots [D]a_k\# \Rightarrow \#[i_R]y\# \Rightarrow \#w_2\# \Rightarrow \dots \Rightarrow \#w_s\#$$

where $w = \gamma(w_s)$.

We will describe a derivation of w in G. According to the construction $\gamma(\alpha)$ and y are axioms of G. Assume that w_k contains returning agent for some k and $\gamma(w_k)$ is in $L(G)$. String w_2 is the first with that property. According to the construction of the BPEG system Σ the derivation follows in that way: The returning agent B_{i_R} moves to the left side of the environment and changes itself into the checking agent B_{i_C}. Checking

agent searches the environment for the symbol a_i. If the symbol is not present in the environment, then the checking agent will rewrite itself into the returning agent B_{j_R}, where $j \in \varphi(i)$. If the symbol a_i is in the environment, then the agent B_{i_C} rewrites itself into the agent B_i. The agent B_i searches the environment for the symbol a_i. In all the steps for derived word $\#w_r\#$ we have $\gamma(w_r) = \gamma(w_k)$ so it is in $L(G)$. The derivation does not change the environment, until the rule $[i]a_i \to [r_R]w_i$ of the simulation agent is used for $r \in \sigma(i)$. This take place exactly when $\gamma(w_r)$ fulfill the conditions for using the rule $(r : a \to w, \sigma(r), \varphi(r))$ in G. Hence the simulating agent rewrites symbol a_i and produces returning agent in the step

$$\#y_1[i]a_iy_2\# \Rightarrow \#y_1[r_R]w_iy_2\#.$$

This derivation step corresponds the derivation step

$$y_1a_iy_2 \Rightarrow y_1w_iy_2$$

in the grammar G. The derivation continues in the same way as it was described above, so it holds $L(\Sigma, \#\alpha\#) \subseteq L(G)$. Hence $L(\Sigma, \#\alpha\#) = L(G)$. □

Lemma 4.2. The family of languages generated by the pure matrix context-free grammars with appearance checking is a subset of the family of languages generated by the bordered positioned eco-grammar systems.
$\mathcal{L}(pM, CF, ac) \subseteq \mathcal{L}(BPEG)$

P r o o f. Consider a pure matrix context-free grammar with appearance checking $G = (V, M, S, F)$, where $M = \{m_1, m_2, \ldots, m_n\}$ is a finite set of finite sequences of productions, $m_i : (a_{i,1} \to w_{i,1}, a_{i,2} \to w_{i,2}, \ldots, a_{i,r_i} \to w_{i,r_i})$, $a_{i,j} \in V$, $w_{i,j} \in V^*$, $1 \leq i \leq n$, $1 \leq j \leq r_i$.

We construct a BPEG system $\Sigma = (V, \#, N_B, E, B_I, B_D, B_{(1,1)}, \ldots, B_{(1,r_1)}, \ldots, B_{(n,1)}, \ldots, B_{(n,r_n)}, B_{(1,1)_R} \ldots, B_{(n,r_n)_R}, B_{(1,1)_C} \ldots, B_{(n,r_n)_C})$, where $n = |M|$ and r_i is the number of productions in $m_i, 1 \leq i \leq n$, such that $L(\Sigma, \#\alpha\#) = L(G)$ for some axiom $\#\alpha\#$.

Alphabet V of the BPEG system Σ is identical with that of the pure grammar G. $E = (V \cup \{\#\}, \{a \to a : a \in V \cup \{\#\}\})$ is $0L$ scheme of BPEG system Σ.

Σ has five types of agents: the initiating agent B_I, the deleting agent B_D, and for each the production $m_{(i,j)} \in M$ of the grammar G we define: the returning agent $B_{(i,j)_R}$, the checking agent $B_{(i,j)_C}$, the simulating agent $B_{(i,j)}$. The set of corresponding agents positions is $N_B = \{[D], [I]\} \cup \{[i, j] : 1 \leq i \leq n, 1 \leq j \leq r_i\} \cup \{[(i, j)_R] : 1 \leq i \leq n, 1 \leq j \leq r_i\} \cup \{[(i, j)_C] : 1 \leq i \leq n, 1 \leq j \leq r_i\}$.

Agents are specified as follows:

- *Initiating agent* $B_I = ([I], Q_I)$, where
  $Q_I = \{\ [I]a \to [(i, 1)]y : a \in V, 1 \leq i \leq n, y \in S\ \}$.
- *Deleting agent* $B_D = ([D], \{[D]a \to \varepsilon :\ a \in V\})$.

For $a_{i,j} \to w_{i,j}$ the jth production of the ith matrix of grammar G we define three agents:

- *Returning agent* $B_{(i,j)_R} = ([(i,j)_R], Q_{(i,j)_R})$ where
  $Q_{(i,j)_R} = \{b[(i,j)_R] \to [(i,j)_R]b : b \in V\} \cup \{\#[(i,j)_R] \to \#[(i,j)_C]\}$.
  It is able to move left and on the leftmost position it changes itself to the checking agent.

- *Checking agent* $B_{(i,j)_C} = ([(i,j)_C], Q_{(i,j)_C})$, where $Q_{(i,j)_C}$ depends on whether the rule $a_{i,j} \to w_{i,j}$ occurs in the set F or not.
  For $a_{i,j} \to w_{i,j} \notin F$ we have
  $Q_{(i,j)_C} = \{[(i,j)_C]b \to b[(i,j)_C] : b \in (V \setminus \{a_{i,j}\})\} \cup \{a_{i,j}[(i,j)_C] \to a_{i,j}[(i,j)]\}$.
  For $a_{i,j} \to w_{i,j} \in F$ furthermore
  $Q_{(i,j)_C} = Q_{(i,j)_C} \cup \{[(i,j)_C]\# \to [(k,l)_R]\#\}$,
  where either $k = i$ and $l = j + 1$ for $1 \leq j < r_i$, or $1 \leq k \leq n$ and $l = 1$ for $j = r_i$.
  Checking agent moves to the right and it looks for the symbol $a_{i,j}$. It can changes itself into the simulating agent.

- *Simulating agent* $B_{(i,j)} = ([(i,j)], Q_{(i,j)})$ where
  $Q_{(i,j)} = \{[(i,j)]a_{i,j} \to [(k,l)_R]w_{i,j}\} \cup \{[(i,j)]b \to b[(i,j)] : b \in V\} \cup \{c[(i,j)] \to [(i,j)]c : c \in V\}$ for $1 \leq j \leq r_i, 1 \leq i \leq n$, where either $k = i$ and $l = j + 1$ for $1 \leq j < r_i$, or $1 \leq k \leq n$ and $l = 1$ for $j = r_i$.
  Simulating agent moves in the environment and changes one of the occurrences of the symbol $a_{i,j}$ in the environment into $w_{i,j}$. At the same time it changes itself to the returning agent of the next production used in the derivation of G.

$\alpha = [I]a_1[D]a_2 \dots [D]a_k$, where $a_1a_2 \dots a_k$ is one of the words from set S of the axioms of the grammar G.

We prove $L(G) = L(\Sigma, \#\alpha\#)$ by showing both set inclusions.

First we call attention to the fact that by the construction of Σ its environment is determined by the non-evolving 0L scheme $E = (V \cup \{\#\}, \{a \to a : a \in V \cup \{\#\}\})$ and all changes in the derivation are done by the agents.

Inclusion $L(G) \subseteq L(\Sigma, \#\alpha\#)$:

Let $y \Rightarrow x_1 \Rightarrow \dots \Rightarrow x_n$ be the derivation in G, $y \in S$.

In the first derivation step of the BPEG system, each deleting agent deletes itself and the symbol on its right hand side and the initiating agent rewrites neighbouring symbol by one of the axioms of the grammar G with returning agent of the first production of an arbitrary matrix of the grammar G. The first derivation step of the BPEG system corresponds to the choice of the axiom and the first rule of the matrix of productions of the grammar G

$$\#[I]a_1[D]a_2 \dots [D]a_k\# \Rightarrow \#[(i,1)_R]y\#.$$

Assume in general that $x_{i-1} \Rightarrow_{m_{i,j}} x_i$ is derivation in G.

Returning agent moves to the left side of the environment and it rewrites itself into checking agent in the next derivation steps.

$\#y_1[(i,j)_R]y_2\# \Rightarrow^* \#[(i,j)_R]x_{i-1}\# \Rightarrow \#[(i,j)_C]x_{i-1}\#$, where $x_y y_2 = x_{i-1}$

Checking agent moves itself to the right side and looks for the symbol which corresponds to the left side of the production which is the agent simulating. If the checking agent finds the symbol then it will rewrite itself into the simulating agent of the production it simulates.

$$\#y_1[(i,j)_C]a_{i,j}y_2\# \Rightarrow \#y_1[(i,j)]a_{i,j}y_2\#, \text{ where } y_1a_{i,j}y_2 = x_{i-1}$$

If the symbol $a_{i,j}$ is not present in the environment and the simulated production is in the set F then the checking agent will rewrite itself into returning agent of the next production in the matrix. Otherwise the derivation is blocked.

$$\#[(i,j)_C]x_{i-1}\# \Rightarrow^* \#x_{i-1}[(i,j)_C]\# \Rightarrow \#x_{i-1}[(k,l)_R]\#,$$

where either $k = i$ and $l = j + 1$ for $1 \leq j < r_i$, or $1 \leq k \leq n$ and $l = 1$ for $j = r_i$.

Simulating agent looks for the symbol to rewrite according to the production it simulates. (Agent does not have to rewrite the first found matching symbol). Once the agent decides to use the rewriting rule simulating the corresponding production of matrix of grammar G, it rewrites found matching symbol in the same way as the production of matrix of grammar G and replaces itself with another returning agent simulating next production used in matrix grammar G. Agent simulating the last production of the matrix after rewriting matching symbol according to the production it simulates changes itself into one of the returning agent simulating first production of the any matrix of grammar G. That simulates choice of the next matrix after applying all the productions of right simulating productions of matrix.

$$\#y_1[(i,j)]a_{i,j}y_2\# \Rightarrow \#y_1[(k,l)_R]w_{i,j}y_2\#,$$

where either $k = i$ and $l = j + 1$ for $1 \leq j < r_i$, or $1 \leq k \leq n$ and $l = 1$ for $j = r_i, y_1a_{i,j}y_2 = x_{i-1}$ and $y_1w_{i,j}y_2 = x_i$.

We obtain in each derivation step of the BPEG system Σ the same word as by the pure matrix context-free grammar G, hence $L(G) \subseteq L(\Sigma, \alpha)$.

Inclusion $L(\Sigma, \#\alpha\#) \subseteq L(G)$:

In the first derivation step of the BPEG system Σ, each deleting agent deletes itself and the symbol on its right-hand side and the initiating agent rewrites its neighbouring symbol to word $w_0 \in S$ with returning agent of the first production of matrix i, $(i,1)_R$, $1 \leq i \leq n$.

$$\#[I]a_1[D]a_2 \ldots [D]a_k\# \Rightarrow \#[(i,1)_R]w_0\#.$$

Corresponding derivation by the grammar G is:

$$w_0 \Rightarrow^* w_0.$$

According to the construction of the BPEG system Σ the derivation follows in that way: The returning agent $B_{(i,j)_R}$ moves to the left side of the environment and changes itself into the checking agent $B_{(i,j)_C}$. The checking agent searches the environment for the symbol $a_{i,j}$. If the symbol is not present in the environment and production $a_{i,j} \to w_{i,j} \in F$, then checking agent will rewrite itself into the returning agent $B_{(k,l)_R}$, where $k = i$ and $l = j + 1$ for $1 \leq j < r_i$, $1 \leq k \leq n$ and $l = 1$ for $j = r_i$. If the symbol $a_{i,j}$ is in the environment, then the agent $B_{(i,j)_C}$ rewrites itself into the agent $B_{(i,j)}$. The agent $B_{(i,j)}$ search the environment for the symbol $a_{i,j}$. All these derivation steps do not change the environment, hence the corresponding derivation by the grammar G is:

$$y \Rightarrow^* y.$$

The simulating agent rewrites symbol $a_{i,j}$ and itself as follows:

$$\#y_1[(i,j)]a_{i,j}y_2\# \Rightarrow \#y_1[(k,l)_R]w_{i,j}y_2\#,$$

where either $k = i$ and $l = j + 1$ for $1 \leq j < r_i$, or $1 \leq k \leq n$ and $l = 1$ for $j = r_i$. The following derivation corresponds to this derivation step by the grammar G:

$$y_1a_{i,j}y_2 \Rightarrow y_1w_{i,j}y_2.$$

The derivation continues in the same way as it was described above, so it holds

$$L(\Sigma, \#\alpha\#) \subseteq L(G).$$

We proved both inclusions, hence

$$L(\Sigma, \#\alpha\#) = L(G).$$

□

Lemma 4.3. The family of languages generated by the pure random context context-free grammars with appearance checking is a subset of the family of languages generated by the bordered positioned eco-grammar systems.
$\mathcal{L}(pRC, CF, ac) \subseteq \mathcal{L}(BPEG)$

P r o o f . Consider a pure random context context-free grammar with appearance checking $G = (V, P, S)$, where $P = \{(a_i \to w_i, Q_i, R_i) : 1 \leq i \leq n\}$, $r_i = |Q_i|$. We construct BPEG system $\Sigma = (V, \#, N_B, E, B_I, B_D, B_{PC}, B_{1_R}, \ldots, B_{n_R}, B_{1_B}, \ldots, B_{n_B}, B_{1,1}, \ldots, B_{n,r_n}, B_{c(1,1)}, \ldots, B_{c(n,r_n)}, B_{1_C}, \ldots, B_{n_C}, B_{1_S}, \ldots, B_{n_S})$, with axiom $\#\alpha\#$ which generates the same language as G, i. e. $L(\Sigma, \#\alpha\#) = L(G)$ for which $\alpha = [I]a_1[D]a_2 \ldots [D]a_k$, where $a_1 a_2 \ldots a_k$ is is one of the words from the set of axioms S of G.

The alphabet V of the environment is identical with the alphabet of the grammar G.

The environment of Σ is determined by the stable 0L scheme $E = (V \cup \{\#\}, \{a \to a : a \in V \cup \{\#\}\})$. Agents of BPEG system Σ correspond to the rules of the grammar G.

BPEG system Σ contains following types of agents: initiating agent B_I, deleting agent B_D, production choosing agent B_{PC}, production checking agents B_{i_B},B_{i_R}, corresponding to the set R_i, production checking agents $B_{i,j}$, $B_{c(i,j)}$, corresponding to the set Q_i and production simulating agents B_{i_S}.

- *Initiating agent* $B_I = ([I], \{[I]a \to [PC]y : a \in V, \quad y \in S\})$.
- *Deleting agent* $B_D = ([D], \{[D]a \to \varepsilon : a \in V\})$.
- *Production choosing agent* $B_{PC} = ([PC], Q_{PC})$ where $Q_{PC} = \{[PC]a \to [i_B]a : a \in (V \cup \{\#\}), 1 \leq i \leq n\}$ chooses one of the productions to be applied by $[i_B]$.

The axiom $\#\alpha\# = \#[I]a_1[D]a_2 \ldots [D]a_k\#$ is rewritten to one of the axioms of the grammar G and to the production choosing agent. The first two derivation steps of the BPEG system Σ are

$$\#[I]a_1[D]a_2 \ldots [D]a_k\# \Rightarrow \#[PC]y\# \Rightarrow \#[i_B]y\#.$$

Production checking agents and production simulating agents of the BPEG system Σ are related to the productions of the grammar G. Assume that the ith production $(a_i \to w_i, Q_i, R_i)$ of G,$1 \leq i \leq n$ has the set Q_i indexed so that a_i is the last symbol in Q_i for $a_i \in Q_i$. So $Q_i = \{a_{i,j} : 1 \leq j \leq r_i\}$, where $a_{i,j} \neq a_i$ for all $j < r_i$ and $a_{i,r_i} = a_i$ in the case that $a_i \in Q_i$.

- *Production checking agents for conditions* R_i, $1 \leq i \leq n$:
  - $B_{i_B} = ([i_B], Q_{i_B})$, where $Q_{i_B} = \{a[i_B] \to [i_B]a : a \in V\} \cup \{\#[i_B] \to \#[i_R]\}$,

– $B_{i_R} = ([i_R], Q_{i_R})$, where
$Q_{i_R} = \{[i_R]a \to a[i_R] : a \in (V \setminus R_i)\} \cup \{[i_R]\# \to [i,1]\#\}$.

Agent B_{i_B} moves to the left # and changes to checking agent B_{i_R}. Agent B_{i_R} moves from the left # to the right and reaches # in the case that none of the letters from R_i is in the current string. Agent changes to the checking agent for conditions Q_i.

- *Production checking agents for conditions* Q_i, where $Q_i = \{a_{i,j} : 1 \leq j \leq r_i\}$, $1 \leq i \leq n$:

  – $B_{i,j} = ([i,j], Q_{i,j})$, where
  $Q_{i,j} = \{a[i,j] \to [i,j]a : a \in V\} \cup \{\#[i,j] \to \#[c(i,j)]\}$ and

  – $B_{c(i,j)} = ([c(i,j)], Q_{c(i,j)})$, where
  $Q_{c(i,j)} = \{[c(i,j)]a \to a[c(i,j)] : a \in V \setminus \{a_{i,j}\}\} \cup$
  $\{[c(i,j)]a_{i,j} \to a_{i,j}[i,j+1]\}$ for $j \neq r_i$.
  $Q_{c(i,r_i)} = \{[c(i,r_i)]a \to a[c(i,r_i)] : a \in V\} \cup \{[c(i,r_i)]a_{i,r_i} \to a_{i,r_i}[i_S]\}$ for $a_i \notin Q_i$.
  $Q_{c(i,r_i)} = \{[c(i,r_i)]a_{r_i} \to a_{r_i}[i_C]\} \cup \{[c(i,r_i)]a \to a[c(i,r_i)] : a \in V \setminus \{a_{r_i}\}\}$ for $a_i \in Q_i$.

  – $B_{i_C} = ([i_C], Q_{i_C})$, where
  $Q_{i_C} = \{[i_C]a_i \to [i_S]a_i\} \cup \{[i_C]a \to a[i_C] : a \in V \setminus \{a_i\}\}$.

The agent identifier $[i,j]$ moves to the left # and it changes itself to the $[c(i,j)]$, which moves to the right and changes itself to $[i,j+1]$ for $j \neq r_i$ when it meets $a_{i,j}$. For $j = r_i$ the $[c(i,r_i)]$ changes itself directly to $[i_S]$ in the case $a_i \notin Q_i$ or through $[i_C]$ for $a_i \in Q_i$.

- *Simulating agents* $B_{i_S} = ([i_S], Q_{i_S})$ for $1 \leq i \leq n$
  $Q_{i_S} = \{a_i[i_S] \to w_i[PC]\} \cup \{a[i_S] \to [i_S]a, [i_S]a \to a[i_S] : a \in V\}$.

Simulating agent searches the environment for the symbol it has to rewrite to simulate the production of the grammar G. (Agent does not have to rewrite the first found matching symbol). Once the agent uses the rewriting rule simulating corresponding production of grammar G, it replaces itself with production choosing agent.

We prove $L(\Sigma, \#\alpha\#) = L(G)$ by verifying both inclusions $L(G) \subseteq L(\Sigma, \#\alpha\#)$ and $L(\Sigma, \#\alpha\#) \subseteq L(G)$.

Assume that $w \in L(G)$ for pure random context context-free grammar with appearance checking $G = (V, P, S)$. This means there is the derivation

$$y = u_1 \Rightarrow \ldots u_i \Rightarrow u_{i+1} \Rightarrow \cdots \Rightarrow^* u_n = w$$

for some $y \in S$ in G. We have to prove that $w \in L(\Sigma, \#\alpha\#)$, ie. we have to find a derivation

$$\#[I]a_1[D]a_2 \ldots [D]a_k\# \Rightarrow \#[PC]y\# \Rightarrow^* \#u\#$$

in Σ such that $\gamma(u) = w$. Evidently $\gamma(\alpha) \in S$ and $\gamma([PC]y) = y \in S$. It is sufficient to show that to each derivation step $u_i \Rightarrow u_{i+1}$, $1 \leq i \leq n-1$ there is a corresponding derivation

$$\#\alpha_i\# \Rightarrow^* \#\alpha_{i+1}\#$$

in Σ such that $\gamma(\alpha_i) = u_i$ and $\#\alpha_i\# = \#\beta_i[PC]\delta_i\#$.

Let the production $(a_j \to w_j, Q_j, R_j)$ be used in the derivation step $u_i \Rightarrow u_{i+1}$ in G. So $u_i = x_i a_j z_i$, $u_{i+1} = x_i w_j z_i$, no letter from R_j occurs in $x_i z_i$ and all letters from Q_j occur in $x_i z_i$. Corresponding derivation in Σ is

$$\#\alpha_i\# = \#\beta_i[PC]\delta_i\# \Rightarrow^* \#\beta_i[i_B]\delta_i\# \Rightarrow^* \#[i_R]\beta_i\delta_i\#$$
$$\Rightarrow^* \#\beta_i\delta_i[i_R]\# \Rightarrow \#\beta_i\delta_i[i,1]\# \Rightarrow^* \#[i,1]\beta_i\delta_i\# \Rightarrow \#[c(i,1)]\beta_i\delta_i\#$$

For $a_i \notin Q_i$ we get:

$$\Rightarrow^* \#x_i[c(i,r_i)]a_i z_i\# \Rightarrow \#x_i[i_S]a_i z_i\#$$

For $a_i \in Q_i$ we get:

$$\Rightarrow^* \#x_i[i_C]a_i z_i\# \Rightarrow \#x_i[i_S]a_i z_i\#$$

The derivation continues for both cases:

$$\Rightarrow^* \#x_i a_i[i_S]z_i\# \Rightarrow \#x_i w_i[PC]z_i\#.$$

This gives $L(G) \subseteq L(\Sigma, \#\alpha\#)$.

To prove that $L(\Sigma, \#\alpha\#) \subseteq L(G)$ we assume that $w \in L(\Sigma, \#\alpha\#)$ for the BPEG system Σ. This means there is a derivation

$$\#\alpha\# \Rightarrow^* \#u\#$$

in Σ such that $\gamma(u) = w$. To show that $w \in L(G)$ we have to find corresponding derivation of w in G. First we call attention to the fact that by the construction of Σ its environment is determined by the non-evolving 0L scheme $E = (V \cup \{\#\}, \{a \to a : a \in V \cup \{\#\}\})$ and all changes in the derivation are done by the agents. Moreover derivation

$$\#\alpha\# \Rightarrow^j \#u_j\#$$

gives u_j with exactly one agent identifier for $j \geq 1$.

Let

$$\#u_t\# \Rightarrow \#u_{t+1}\#$$

be a derivation step in the derivation

$$\#\alpha\# \Rightarrow^* \#u_t\#$$

for $u_t \neq \alpha$. Then $\gamma(u_t)$ and $\gamma(u_{t+1})$ can differ only for simulating agents. In the case that the agent in $\#u_t\#$ is a production choosing agent or a production checking agent it holds $\gamma(u_t) = \gamma(u_{t+1})$. So it is sufficient to verify that $\gamma(u_{t+1}) \in L(G)$ for

$$\#u_t\# \Rightarrow \#u_{t+1}\#,$$

where $u_t = x a_i[i_S]z$ and $\gamma(u_t) \in L(G)$. From the construction of Σ it follows that the only way to derive $\#x[i_S]z\#$ in Σ is the following:

$$\#\alpha\# \Rightarrow^* \#x'[PC]z'\# \Rightarrow \#x'[i_B]z'\# \Rightarrow^* \#[i_B]x'z'\# \Rightarrow \#[i_R]x'z'\# \Rightarrow^* \#x'z'[i_R]\#$$

(the derivation guarantees that $x'z'$ does not contain letters from R_i). The derivation continues as follows:

$$\Rightarrow \#x'z'[i,1]\# \Rightarrow^* \#[i,1]x'z'\# \Rightarrow \#[c(i,1)]x'z'\# \Rightarrow^* \#x''[c(i,1)]a_{i,1}z''\# \Rightarrow$$
$$\#x''a_{i,1}[i,2]z''\#$$

(this gives $x^{'}z^{'} = x^{''}a_{i,1}z^{''}$ for $a_{i,1}$ from Q_i). The derivation continues testing occurrences of all other letters from Q_i in $x^{'}z^{'}$.

$$\Rightarrow^* \#y_1[c(i,r_i)]a_{i,r_i}y_2\# \Rightarrow^+ \#y_1a_{i,r_i}[i_S]y_2\#$$

(the last configuration guarantees that $y_1a_{i,r_i}y_2$ contains all letters from Q_i)

$$\Rightarrow^* \#xa_i[i_S]z\# \Rightarrow \#xw_i[PC]z\#.$$

Evidently $\gamma(u_{t+1}) = \gamma(xw_i[PC]z) = xw_iz$. According to the notes in the derivation $x^{'}z^{'} = xa_iz$ and $xa_iz \Rightarrow xw_iz$ in G gives $xw_iz \in L(G)$.

In this way the BPEG system Σ simulates derivations of the pure random context context-free grammar G and vice versa, hence $L(\Sigma, \#\alpha\#) = L(G)$. □

5. MAIN RESULTS

In the previous section we have shown that all the families of languages generated by the pure regulated context-free grammars with appearance checking are subsets of the family of BPEG languages. All these subsets are proper subsets of the family of BPEG languages. For the proof of this statements we will use languages from the second section of this paper.

Theorem 5.1. The family of languages generated by the pure programmed context-free grammars with appearance checking is the proper subset of the family of languages generated by the bordered positioned eco-grammar systems.
$\mathcal{L}(pP, CF, ac) \subsetneq \mathcal{L}(BPEG)$

Proof. According to the lemma 4.1 it holds: $\mathcal{L}(pP, CF, ac) \subseteq \mathcal{L}(BPEG)$. According to the lemma 4.2 it holds: $\mathcal{L}(pM, CF, ac) \subseteq \mathcal{L}(BPEG)$. According to the [4] for the language L_1 from the proposition 2.1 it holds: $L_1 \in \mathcal{L}(pM, CF)$ and $L_1 \notin \mathcal{L}(pP, CF, ac)$, hence: $\mathcal{L}(pP, CF, ac) \subsetneq \mathcal{L}(BPEG)$. □

Theorem 5.2. The family of languages generated by the pure matrix context-free grammars with appearance checking is the proper subset of the family of languages generated by the bordered positioned eco-grammar systems.
$\mathcal{L}(pM, CF, ac) \subsetneq \mathcal{L}(BPEG)$

Proof. According to the lemma 4.2 it holds: $\mathcal{L}(pM, CF, ac) \subseteq \mathcal{L}(BPEG)$. According to the lemma 4.1 it holds: $\mathcal{L}(pP, CF, ac) \subseteq \mathcal{L}(BPEG)$. According to the [4] for the language L_2 from the proposition 2.1 it holds: $L_2 \in \mathcal{L}(pP, CF)$ and $L_2 \notin \mathcal{L}(pM, CF, ac)$, hence: $\mathcal{L}(pM, CF, ac) \subsetneq \mathcal{L}(BPEG)$. □

Theorem 5.3. The family of languages generated by the pure random context context-free grammars with appearance checking is the proper subset of the family of languages generated by the bordered positioned eco-grammar systems.
$\mathcal{L}(pRC, CF, ac) \subsetneq \mathcal{L}(BPEG)$

P r o o f . According to the lemma 4.3 it holds: $\mathcal{L}(pRC, CF, ac) \subseteq \mathcal{L}(BPEG)$. According to the lemma 4.2 it holds: $\mathcal{L}(pM, CF, ac) \subseteq \mathcal{L}(BPEG)$. According to the [4] for the language L_1 from the proposition 2.1 it holds: $L_1 \in \mathcal{L}(pM, CF)$ and $L_1 \notin \mathcal{L}(pRC, CF, ac)$, hence: $\mathcal{L}(pRC, CF, ac) \subsetneq \mathcal{L}(BPEG)$. □

Corollary. The family of languages generated by pure regulated context-free grammars with appearance checking is a proper subset of the family of BPEG languages.

6. CONCLUSION

The main result of the paper formulated in Section 5 states that families of pure programmed, matrix and random context context-free languages with appearance checking are properly included in the family of $BPEG$ languages. The question whether pure regulated languages with appearance checking are included in the PEG languages remains opened. Neither was studied the relation between PEG and BPEG systems.

ACKNOWLEDGEMENT

Article has been made in connection with project IT4Innovations Centre of Excellence, reg. no. CZ.1.05/1.1.00/02.0070 supported by Research and Development for Innovations Operational Programme financed by Structural Founds of Europe Union and from the means of state budget of the Czech Republic.

Research was also supported by the SGS 5/2010 Project of the Silesian university Opava.

(Received August 10, 2011)

REFERENCES

[1] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, and Gh. Păun: Grammar Systems – A Grammatical Approach to Distribution and Cooperation. Gordon and Breach, London, 1994.

[2] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová, and Gh. Păun: Eco-grammar systems. A grammatical framework pro studying lifelike interactions. Artificial Life *3*, (1997), 1–28.

[3] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová, and Gh. Păun: Eco(grammar) systems – A preview. In: Cybernetics a Systems '94 (R. Trappl, ed.), World Scientific, Singapore 1994, pp. 941–948.

[4] J. Dassow and Gh. Păun: Regulated Rewriting in Formal Language Theory. Akademie-Verlag, Berlin, 1989.

[5] J. Dassow, Gh. Păun, G. Rozenberg: Grammar systems. In: Handbook of Formal Languages, Vol. 2 (G. Rozenberg and A. Salomaa, eds.), Springer-Verlag, Berlin 1997, pp. 155–214.

[6] J. Dassow, Gh. Păun, and A. Salomaa: Grammars with controlled derivations. In: Handbook of Formal Languages, Vol. 2 (G. Rozenberg, A. Salomaa, eds.), Springer-Verlag, Berlin 1997, pp. 101–154.

[7] L. Kari, G. Rozenberg, and A. Salomaa: L-systems. In: Handbook of Formal Languages. Vol.1 (G. Rozenberg, A. Salomaa. eds.), Springer-Verlag, Berlin 1997, pp. 253–324.

[8] J. Kelemen and A. Kelemenová: A subsumption architecture for generative symbol systems. In: Cybernetics and System Reseach '92 (R. Trappl ed.), World Scientific, Singapore 1992, pp. 1529–1536.

[9] A. Kelemenová and M. Langer: Positioned agents in eco-grammar systems. Internat. J. Found. Comput. Sci. *22*, (2011), 237–246.

[10] M. Langer: Agents placed in the environment of eco-grammar systems – Positioned eco-grammar systems. In: Pre-Procs. 1st Doctoral Workshop on Mathematical a Engineering Methods in Computer Science (M. Češka et al., eds.), FI MU, Brno 2005, pp. 31–37.

[11] C. Martin-Vide and Gh. Păun: New topics in colonies theory. Grammars *1*, (1999), 209–323.

[12] C. Martin-Vide and Gh. Păun: PM-colonies. Comput. Artif. Intell. *17*, (1998), 553–582.

[13] Gh. Păun and A. Salomaa: Families generated by grammars and L systems. In: Handbook of Formal Languages, Vol.1 (G. Rozenberg, A. Salomaa, eds.), Springer, Berlin 1997, pp. 811–859.

Miroslav Langer, Institute of Computer Science and Research Institute of the IT4 Inovations Centre of Excelence, Silesian University, Bezručovo nám. 13, 746 01 Opava. Czech Republic.
e-mail: miroslav.langer@fpf.slu.cz

Alica Kelemenová, Institute of Computer Science and Research Institute of the IT4 Inovations Centre of Excelence, Silesian University, Bezručovo nám. 13, 746 01 Opava. Czech Republic.
e-mail: kelemenova@fpf.slu.cz