ON POSITIONED ECO-GRAMMAR SYSTEMS AND PURE GRAMMARS OF TYPE 0

Miroslav Langer[*], *Alica Kelemenová*[*]

Abstract: In this paper we extend our results given in [5] where we compared PEG systems with pure regulated context-free grammars (see [3]). We will show that the family of languages generated by the pure grammars of type 0 is a proper subclass of the family of languages generated by positioned eco-grammar systems. We present a way how to coordinate parallel behavior of agents with one-sided context in a PEG system in order to simulate the derivation step in a pure grammar of type 0 determined by a single rule which replaces an arbitrarily long string by another one. Related results concerning PEG systems and pure languages can be found in [6].

Key words: *Eco-grammar systems, positioned eco-grammar systems, pure grammars*

Received: July 15, 2012
Revised and accepted: March 28, 2013

1. Introduction

Eco-grammar systems represent grammar systems which were motivated by the behavior of ecosystems. They were introduced in 1994 (see [2]) in order to describe evolving environment and community of agents together with the interplay between them. We refer to the paper [1] for all introductory informal and formal comments and results connected with eco-grammar systems.

Positioned eco-grammar systems (PEG systems, for short) were introduced in [8] as a variant of the eco-grammar systems. Motivation for introducing the PEG systems and their main difference from the original eco-grammar systems is that we focus on the embodiment of the agents and their presence in the environment. Actions of the agent are strictly determined by its position in the environment, by the symbol located immediately next to the position of the agent.

PEG systems combine approaches known from the study of the above mentioned eco-grammar systems and PM-colonies (see [10]). The environment of the PEG

[*]Miroslav Langer, Alica Kelemenová
Institute of Computer Science and Research Institute of the IT4 Innovations Centre of Excellence, Silesian University, Bezručovo nám. 13, 746 01 Opava, miroslav.langer@fpf.slu.cz, alica.kelemenova@fpf.slu.cz

system is represented by $0L$ scheme like in the eco-grammar systems. Similarly as in PM-colonies, the presence of each agent in the environment is given by its identifier. Agent rules allow replacing the agent identifier together with one of its neighbouring environmental symbol by a string composed of environmental symbols and agent identifiers.

PEG systems bring new view to the investigation of the interplay between the community of agents and the environment. Our approach allows studying local changes in evolving environment caused by agents. Moreover, the position of an agent is strictly given by the special symbol and we are able to predict its behavior and control the evolution of the environment as well. Actions of agents have priority over the development of the environment and they are synchronized in a totally parallel way.

As usual, we use a formal language for representation of all the states of the PEG systems which are reachable from the starting string, the axiom. A natural and basic task is to determine the position of the family of PEG languages in known hierarchies of languages and compare it with other language classes.

In [5] we compared generative power of the PEG systems and generative power of the pure regulated context-free grammars (see [3]). Main results of the paper state that families of pure programmed context-free languages, pure matrix context-free languages and pure random context context-free languages are properly included in the family of PEG languages. We did not consider appearance checking in the regulated context-free grammars in [5].

In [6] we looked for the relation between the regulated pure context-free families with appearance checking and the family of PEG languages. To simulate appearance checking we have to modify slightly a PEG system by adding boundary markers to get BPEG systems. After such a modification we proved that each of the families of pure programmed context-free languages with appearance checking, pure matrix context-free languages with appearance checking, and pure random context context-free languages with appearance checking are properly included in the family of BPEG languages.

The goal of the present paper is to compare family of languages generated by the pure grammars of type 0 and PEG languages. While in previous papers the advantages of the context-free rules were used, here we emphasize that we can coordinate behavior of the agents with one-sided single letter context and simulate the derivation of the grammar with rules rewriting arbitrarily long substrings of the environment. Thus we show that the family of languages generated by the pure grammars of type 0 constitutes a proper subclass of the family of languages generated by the positioned eco-grammars systems.

The structure of the paper is as follows: In Section 2 the reader can find preliminaries on pure grammars of type 0 and their language class. Also Lindenmayer systems are reviewed in this part of the paper. They will be used to describe behavior of the environment of the PEG system.

Definitions of the positioned eco-grammar system are singled out and illustrated in Section 3.

Main theorem is proved in detail in Section 4. Simulation of the derivation process in pure grammar of type 0 by that of PEG system will be realized by several specialized agents. Before we present all the technical details of formal proof we will describe basic ideas of the proof and the roles of individual agents informally.

We assume that the reader is familiar with the formal language theory including Lindenmayer systems (see [7, 11]) and pure grammars (see [3, 4]).

2. Preliminaries on Formal Grammars

In order to introduce main object of this paper, the positioned eco-grammar systems, we recall the notion of Lindenmayer systems which form their substantial parts and also the notion of the pure grammars, another basic notion of the paper.

In the classical Chomsky classification of formal grammars the alphabet of a grammar is divided into two disjoint subsets, terminal and non-terminal symbols. The language generated by Chomsky grammar contains words composed of the terminal symbols only. The intermediate sentential forms, i.e. strings generated as intermediate steps in the derivation, are not included in the language.

If we consider pure grammars, we do not divide the alphabet into subsets of terminal and non-terminal symbols. The language generated by the pure grammar contains all strings obtained from the axiom during the derivation. It means that the language contains also the procedure of generating the language, thus the strings match the sentential forms in the Chomsky classification of grammars. Languages generated by the pure grammars are called pure languages or also the languages of the sentential forms.

Definition 1 *A pure grammar of type 0 is a triplet $G = (V, P, S)$, where*

- *V is a finite nonempty alphabet,*
- *P is a finite set of rules of type $u \to w, u \in V^+, w \in V^*$,*
- *$S \subseteq V^+$ is a finite set of the axioms.*

Definition 2 *A derivation step in a pure grammar of type 0 is a binary relation $\Rightarrow$ defined as follows: $x \Rightarrow y$ for $x \in V^+$, $y \in V^*$, iff $x = z_1uz_2, y = z_1wz_2$ and $u \to w \in P$, $z_1, z_2 \in V^*$.*

Definition 3 *A language defined by the pure grammar G of type 0 is a set $L(G) = \{y : x \Rightarrow^* y, x \in S\}$, where $\Rightarrow^*$ is reflexive and transitive closure of the relation $\Rightarrow$.*

The class of all languages defined by the pure grammars of type 0 is denoted by $\mathcal{L}(pRE)$.

Lindenmayer systems, L-systems for short, are pure parallel grammars. Lindenmayer systems were introduced by Aristid Lindenmayer in 1968 as an attempt to describe growth of the multicellular organisms.

The simplest type of the L systems is the form of L systems with no interaction, the $0L$ systems for short. The alphabet of the $0L$ system is not divided into subsets of terminal and non-terminal symbols as well. The rules of the $0L$ system are context-free.

Definition 4 *A $0L$ system is triplet $G = (V, P, w_0)$, where*

- *V is finite nonempty alphabet,*
- *P is finite nonempty set of rules of the type $a \to \alpha$, where $a \in V$ and $\alpha \in V^*$,*
- *$w_0 \in V^+$ is the axiom of the system.*

Definition 5 *A derivation step in 0L system is a binary relation $\Rightarrow$ defined as follows: $a_1 a_2 \dots a_n \Rightarrow \beta_1 \beta_2 \dots \beta_n$, where:*

- *$a_1, a_2, \dots, a_n \in V$,*
- *$a_1 \to \beta_1, a_2 \to \beta_2, \dots, a_n \to \beta_n \in P$ are rules of the $0L$ system.*

Definition 6 *A language defined by the $0L$ system G is a set $L(G) = \{w \colon w \in V^*, w_0 \Rightarrow^* w\}$, where $\Rightarrow^*$ is reflexive and transitive closure of the relation $\Rightarrow$.*

The class of all languages defined by $0L$ systems ($0L$ languages) is denoted by $\mathcal{L}(0L)$.

A $0L$ scheme is a part of a $0L$ system, where the axiom is not considered, i.e. the pair (V, P). We will use $0L$ schemes to describe behavior of the environment in the eco-grammar systems.

Example 1 *An example of the $0L$ system generating strings of the length of the Fibonacci numbers: $G = (V, P, w_0)$,*

- *$V = \{a, b\}$,*
- *$P = \{a \to ab, b \to a\}$,*
- *$w_0 = b$.*

We show several derivation steps in G: $b \Rightarrow a \Rightarrow ab \Rightarrow aba \Rightarrow abaab \Rightarrow abaababa \Rightarrow abaababaabaab \Rightarrow abaababaabaababaababa \dots$

3. Positioned Eco-Grammar Systems

In the present section we will deal with the positioned eco-grammar systems introduced in [8]. They are modification of the earlier introduced eco-grammar systems (see [2, 1]).

Definition 7 *A positioned eco-grammar system (PEG system, for short) of degree m, $m \geq 1$, is an $(m+3)$ – tuple $\Sigma = (V_E, N_B, E, B_1, \dots, B_m)$, where*

- *V_E is a finite nonempty alphabet of the environment,*
- *$N_B = \{[j] : 1 \leq j \leq m\}$ is the set of identifiers of agents, $[j]$ determines positions of the j-th type agent in the environment,*
- *$E = (V_E, P_E)$ is a $0L$ scheme of the environment with alphabet V_E and developmental rules P_E,*

- $B_j = ([j], Q_j)$, *is the* j*-th type agent for* $1 \leq j \leq m$ *and* Q_j *– a set of rules of type* $a[j] \to u$, *or* $[j]a \to u$, *where* $a \in V_E$ *is a symbol marking (left or right) vicinity with the agent* $[j]$ *and* $u \in (V_E \cup N_B)^*$.

Note that the definition requires at least one agent in each positioned eco-grammar system. An agent can arise or die via the rules and its appearance in the environment is given by a symbol from the set N_B.

Definition 8 *A configuration of the PEG system* $\Sigma = (V_E, N_B, E, B_1, \ldots, B_m)$ *is a word* v *in* $(V_E \cup N_B)^*$. *The starting configuration will be called the axiom.*

Derivation step of the *PEG* system will describe following behavior of the system: All agents in the configuration are active and work in parallel. In a derivation step each agent has to rewrite one symbol on its right or left context according to a rule from corresponding Q_j. Otherwise no derivation step is possible. The rest of the symbols (i.e. those not touched by agents) are rewritten by the rules of the environment.

The derivation step of the PEG system is introduced as follows:

Definition 9 *A derivation step of PEG system* $\Sigma = (V_E, N_B, E, B_1, \ldots, B_m)$ *is a binary relation* $\Rightarrow_\Sigma$ *over* $(V_E \cup N_B)^*$, *such that* $w \Rightarrow_\Sigma w'$ *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* $0 \leq k \leq n$.

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

Note that the derivation step is not determined if two agents have the only possibility to rewrite their common neighbouring symbol. Agents are in conflict in a given string and our systems are not able to solve such a conflict. The derivation is blocked. Another case occurs when derivation stops due to absence of rule for the agent in actual context. Completeness of sets Q_j are not required in the definition.

To define the language of *PEG* system we have also to fix the axiom. The language defined by the positioned eco-grammar system and axiom is given by all strings produced by the system from the axiom, ignoring agents identifiers.

Definition 10 *Language defined by PEG system* $\Sigma = (V_E, N_B, E, B_1, \ldots, B_m)$ *and axiom* $w \in (V_E \cup N_B)^*$ *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 positioned eco-grammar systems (*PEG* languages) is denoted $\mathcal{L}(PEG)$.

Example 2 *We present an example of PEG system Σ. Let*
$\Sigma = (\{a,b,c\}, \{[1],[2],[3]\}, (\{a,b,c\}, \{a \to a, b \to b, c \to c\}), B_1, B_2, B_3)$

- $B_1 = ([1], \{a[1] \to aa[2], b[1] \to [1]b, c[1] \to [1]c\})$,
- $B_2 = ([2], \{[2]b \to bb[3]\})$,
- $B_3 = ([3], \{[3]c \to [1]cc, [3]b \to b[3]\})$.

Let $w = a[1]bc$ be the axiom for PEG system Σ.

We show several derivation steps: $a[1]bc \Rightarrow aa[2]bc \Rightarrow aabb[3]c \Rightarrow aabb[1]cc \Rightarrow aab[1]bcc \Rightarrow aa[1]bbcc \Rightarrow aaa[2]bbcc \Rightarrow aaabb[3]bcc \Rightarrow aaabbb[3]cc \Rightarrow aaabbb[1]ccc$.

All the derived strings, where we eliminate positions of the agents, are in the language generated by Σ from the axiom w.

One can verify that the PEG system Σ with $w = a[1]bc$ generates the language $L(\Sigma, a[1]bc) = \{a^n b^n c^n : n \geq 1\} \cup \{a^{n+1} b^n c^n : n \geq 1\} \cup \{a^{n+1} b^{n+1} c^n : n \geq 1\}$.

4. Positioned Eco-Grammar Systems and the Pure Grammars of Type 0

In this section we extend our results given in [5] where we compared PEG systems with pure regulated context-free grammars. Some basic and closure properties of the PEG systems can be found in [8] and [9].

Theorem 1 *The class of languages defined by the pure grammars of type 0 is a proper subclass of the class of the languages defined by the positioned eco-grammar systems $\mathcal{L}(pRE) \subsetneq \mathcal{L}(PEG)$.*

Let us explain the idea of the proof first: Consider pure grammar of type 0 $G = (V, P, S)$. We construct PEG system Σ generating the same language as the grammar G. G starts the computation with one of the finite set of axioms, while the PEG system Σ has to have one axiom. It can be constructed from an arbitrary word $w \in L(G)$, where there is each symbol except one taken by a special deleting agent. The remaining symbol of the axiom is taken by the initializing agent. The only job of each deleting agent is to delete its neighbour symbol and the *initializing agent* generates from the symbol any one of the axioms of grammar G and rewrites itself into *production choosing agent.*

In order to simulate derivation step of the grammar G in PEG system Σ we will consider three sets of the agents. *Countdown deleting agents*, *initializing agent* and the agents simulating the rules of the grammar G, *simulating agents.*

The context of the PEG system agent is given by the only symbol on its left or right side, so the agent can rewrite in one derivation step this only symbol. To simulate the rule of pure grammar of type 0 we have to replace one string (left side of the rule) of an arbitrary length with other string (right side of the rule) in one derivation step. It means that we have to find the sequence of the symbols in the environment first. Then we have to delete the entire string and generate the other one in one derivation step. By the timing of deletion of each symbol of the string we ensure the deletion of the whole string. Every symbol except one is taken by the countdown deleting agent. The last symbol is taken by the agent which generates the right side of the rule. The countdown deleting agents count down the derivation steps to delete all the symbols at the same time. At the same moment the generating agent generates the right side of the rule. Particular types of the countdown deleting agents are denoted by the indexes. The index represents the number of the derivation steps in which the context symbol of the agent will be deleted. The highest index of the countdown deleting agent is one less than the length of the longest left side of all the rules of the grammar G.

Each rule of the grammar G is represented by three subsets of the set of simulating agents. The agents from the first set (*left side checking agents*) ensure verification of occurrence of the left side of the rule. The agents from the second set (*deploying agents*) ensure deploying of the countdown deleting agents by each symbol of the left side of the rule. The agent from the third set (*generating agent*) ensures generating the right side of the rule.

The axiom of the PEG system Σ is constructed from an arbitrary word $w \in L(G)$, where there is each symbol except one taken by the deleting agent. The last symbol is taken by the initializing agent. The *initializing agent* generates one of the axioms of grammar G and rewrites itself into *production choosing agent*. The production choosing agent wanders in the environment and looks for the first symbol of the left side of some rule of grammar G. If the required symbol is found, the production choosing agent rewrites itself into the first left side checking agent. These agents verify whether there is a sequence of the symbols corresponding to the left side of the chosen rule of grammar G. If the required sequence of the symbols is in the environment, the deploying agents deploy countdown deleting agents and the entire sequence is deleted in one derivation step and, at the same time, the right side of the rule together with the production choosing agent is generated. In this way we obtain the same word which can be generated by the grammar G. Detailed proof with all technical details follows.

Proof Consider the pure grammar of type 0 $G = (V, P, S)$, where P is the finite set of rules of type $\alpha \to \beta$. Let $r = |P|$ be the number of the rules and $n = max\{|\alpha| : \alpha \to \beta \in P\}$ be the length of the longest left side of the rules of the grammar G. We assign to each rule integer number from the range $[1, r]$ which is uniquely determining the rule.

We construct the PEG system $\Sigma = (V, N_B, E, B_I, B_{D_1}, \ldots, B_{D_{n-1}}, B_R, B_{R_1^1}, \ldots, B_{R^r_{l_r+1}}, B_{B_1^1}, \ldots, B_{B^r_{l_r+1}})$ where $l_i = |\alpha_i|$ and $\alpha_i \to \beta_i \in P$ is the i-th rule of the grammar, such that $L(\Sigma, \omega) = L(G)$. $E = (V, \{a \to a : a \in V\})$ is $0L$ scheme of the PEG system Σ.

Particular types of agents are defined as follows:

Initializing agent is defined as follows: $B_I = ([I], \{[I]a \to [R]y, a \in V, y \in S\})$. This agent is responsible for choice of the axiom of the grammar G and it generates production choosing agent.

Countdown deleting agents $B_{D_1}, \ldots, B_{D_{n-1}}$ are defined as follows:

$B_{D_i} = ([D_i], \{[D_i]a \to [D_{i-1}]a : a \in V\})$, for $i = 2, \ldots, n-1$ and $B_{D_1} = ([D_1], \{[D_1]a \to \varepsilon : a \in V\})$. These agents are responsible for deleting the whole left side of the rule of the grammar G.

Rule choosing agent: $B_R = ([R], Q_R)$ and $Q_R = \{[R]a \to a[R], a[R] \to [R]a : a \in V\} \cup \{[R]b_i \to [R_i^1]b_i : 1 \leq i \leq r\}$ where $b_i\gamma \to \beta$ is the i-th rule of the grammar G and b_i is the first symbol of its left side, for $1 \leq i \leq r$.

Rule choosing agent wanders in the environment without changing it. Whenever it finds the first symbol of an arbitrary rule of the grammar G, it can rewrite itself into the first left side checking agent of the particular rule. This represents nondeterministic choice of the rule in the derivation of the grammar G.

For each rule of the grammar G we define *simulating agents* containing three parts. Namely *left side checking agents*, *deploying agents* and a *generating agent.*

Left side checking agents have to verify if there is a sequence of the symbols corresponding to the left side of the rule in the environment. If the identification of the left side of the simulated rule is successful, the *deploying agents* ensure deployment of the countdown deleting agents and the generating agent.

The *left side checking agents* are defined as follows: Consider $\alpha_i \to \beta_i$, $\alpha_i = a_1 \dots a_{l_i}$ i-th rule of the grammar G, l_i is the length of its left side and $1 \leq i \leq r, 1 \leq j \leq l_i$.

- $B_{R_i^j} = ([R_i^j], Q_i^j)$ where $Q_i^j = \{([R_i^j]a_j \to a_j[R_i^{j+1}]), ([R_i^j]b \to [R]b); b \in V\}$.
- $B_{R_i^{l_i+1}} = ([R_i^{l_i+1}], Q_i^{l_i+1})$ where $Q_i^{l_i+1} = \{a_{l_i}[R_i^{l_i+1}] \to a_l[B_i^{l_i}]\}$.

The last, $(l_i + 1)$-th agent from the set of left side checking agents is generated if there is sequence of the symbols corresponding to the left side of the simulated rule in the environment. $(l_i + 1)$-th agent rewrites itself into the first deploying agent.

The *deploying agents* are defined as follows: Consider $\alpha_i \to \beta_i$, $\alpha_i = a_1 \dots a_{l_i}$ i-th rule of the grammar G, l_i is the length of its left side and $1 \leq i \leq r, 2 \leq j \leq l_i$.

- $B_{B_i^j} = ([B_i^j], Q_i^j)$ where $Q_i^j = \{(a_j[B_i^j] \to [B_i^{j-1}][D_{j-1}]a_j)\})$. Agent ensures deploying of the countdown deleting agents so that all the symbols $a_1 \dots a_{l_i}$ are deleted and string β_i is generated in one derivation step.

The *generating agent* generate the string β. The agent is defined as follows:

- $B_{B_i^1} = ([B_i^1], Q_i^1)$ where $Q_i^1 = \{(a_1[B_i^1] \to [R]\beta_i\}$. Agent $B_{B_i^1}$ ensures replacement of the first symbol of the left side of the i-th rule by the string β_i; the right side of the rule. After that the agent rewrites itself into the rule choosing agent B_R.

The axiom ω of the PEG system Σ is a string $\omega = [I]a_1[D_1]a_2 \dots [D_1]a_k$ where $a_1a_2 \dots a_k \in L(G)$ is an arbitrary word from the set $L(G)$.

Each deleting agent deletes itself and the symbol on its right side in the first derivation step of the PEG system. At the same time the initializing agent generates from the last symbol an arbitrary axiom from the set S of the grammar G and rewrites itself to the production choosing agent B_R. This behavior of the PEG system simulates choosing of the axiom of the grammar G.

We prove the equality $L(\Sigma, \omega) = L(G)$ by showing both of the set inclusions. Firstly, 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 $L(G) \subseteq L(\Sigma, \omega)$.

At first we show that we can derive an arbitrary axiom of the grammar G. Let $w_0 \in S$ in the grammar G. Corresponding derivation in the PEG system Σ is of the form:
$[I]a_1[D_1]a_2 \dots [D_1]a_k \Rightarrow [R]w_0$ whereas we used the rules $[D_1]a \rightarrow \varepsilon$ and $[I]a_1 \rightarrow [R]w_0$. Evidently $\gamma([R]w_0) = w_0$.

Now we show that if the derivation $w_i \Rightarrow w_{i+1}$, is in the pure grammar G of type 0, then there exists the derivation in the PEG system Σ $w_i^{'} \Rightarrow^* w_{i+1}^{'}$ such that $\gamma(w_i^{'}) = w_i$ and $\gamma(w_{i+1}^{'}) = w_{i+1}$.

- The derivation in the grammar G: $w_{i-1} \Rightarrow w_i$ whereas we used the rule $\alpha_i \rightarrow \beta_i$ on the word $w_{i-1} = x\alpha_i y$, let $\alpha_i = a_1 \dots a_{l_i}$ and $w_i = x\beta_i y$.

- Let the configuration of the PEG system Σ be without loss of generality $[R]x\alpha_i y$, whereas $\gamma([R]x\alpha_i y) = x\alpha_i y = w_{i-1}$. Corresponding derivation in the PEG system Σ is:

  - $[R]xa_1 \dots a_{l_i}y \Rightarrow^{|x|} x[R]a_1 \dots a_{l_i}y \Rightarrow$
    choice of the rule, beginning of verification of the left side of the rule
    $\Rightarrow x[R_i^1]a_1 \dots a_{l_i}y \Rightarrow$
    $\Rightarrow xa_1[R_i^2] \dots a_{l_i}y \Rightarrow \dots \Rightarrow xa_1 \dots [R_i^{l_i}]a_{l_i}y \Rightarrow xa_1 \dots a_{l_i}[R_i^{l_i+1}]y \Rightarrow$
    successful verification of the left side of the rule, deploying countdown deleting agents
    $\Rightarrow xa_1 \dots a_{l_i}[B_i^{l_i}]y \Rightarrow xa_1 \dots \; [B_i^{l_i-1}][D_{l_i-1}]a_{l_i}y \; \dots \Rightarrow$
    countdown deleting agents deployed
    $\Rightarrow xa_1[B_i^1][D_1]a_2 \dots [D_1]a_{l_i}y \Rightarrow x[R]\beta_i y$
    left side of the rule deleted, rule applied.

    Whereas we used the rules:
    $[R]a \rightarrow a[R]$, $[R]a_1 \rightarrow [R_i^2]a_1$, $[R_i^1]a_1 \rightarrow a_1[R_i^2] \; \dots \; [R_i^{l_i}]a_{l_i} \rightarrow a_{l_i}[R_i^{l_i+1}]$, $a_{l_i}[R_i^{l_i+1}] \rightarrow a_{l_i}[B_i^{l_i}]$, $a_{l_i}[B_i^{l_i}] \rightarrow [B_i^{l_i-1}][D_{l_i-1}]a_{l_i} \; \dots \; a_2[B_i^2] \rightarrow [B_i^1][D_1]a_2$, $a_1[B_i^1] \rightarrow [R]\beta_i$, $[D_{l_i-1}]a \rightarrow [D_{l_i-2}]a \; \dots \; [D_2]a \rightarrow [D_1]a$, $[D_1]a \rightarrow \varepsilon$

  It holds $\gamma(x[R]\beta_i y) = x\beta_i y = w_i$.

We show that $L(\Sigma, \omega) \subseteq L(G)$ so we show that for each word w_i derived in the PEG system Σ holds $\gamma(w_i) \in L(G)$.

For the axiom of the PEG system Σ we have:

- According to the definition of the PEG system Σ holds, $\gamma(\omega) = a_1a_2 \dots a_k$ where $a_1a_2 \dots a_k \in L(G)$.

For the first derivation step from the axiom of the PEG system Σ we have:

- Derivation in the PEG system Σ is: $\omega \Rightarrow [R]w_0$ whereas we used the rules $[D_1]a \rightarrow \varepsilon$ and $[I]a_1 \rightarrow [R]w_0$.
  According to the definition it holds $\gamma(\omega) = a_1 \dots a_k \in L(G)$ and $\gamma([R]w_0) = w_0 \in L(G)$ is the axiom of the grammar G.

Now we show that if $w_i \Rightarrow w_{i+1}$ is the derivation in the PEG system Σ, then there is in the pure grammar of type 0 G the derivation $w_i^{'} \Rightarrow^* w_{i+1}^{'}$ such that if $w_i^{'} = \gamma(w_i)$, then $w_{i+1}^{'} = \gamma(w_{i+1})$.

From the definition of the PEG system Σ it implies that only the agents $B_{B_1^1} \dots B_{B_r^1}$ and B_{D_1} can change the environment.

By the acting of the other agents we get the derivation $w_i \Rightarrow w_{i+1}$ where $\gamma(w_i) = \gamma(w_{i+1})$. Corresponding derivation in the grammar G is $\gamma(w_i) \Rightarrow^0 \gamma(w_{i+1})$.

The configurations in which the environment is changed are:

$$w_i = w_{i_1} a_1 [B_i^1][D_1] a_2 \dots [D_1] a_{l_i} w_{i_2},$$

where $w_{i_1}, w_{i_2} \in V^*$ and $a_1 \dots a_{l_i} \in V^+$ is the left side of the i-th rule of the grammar G. In the next derivation step each agent B_{D_1} deletes itself and the symbol on its right side and agent $B_{B_i^1}$ generates the string $[R]\beta_i$. The following configuration is

$$w_{i+1} = w_{i_1} [R] \beta_i w_{i_2}.$$

According to the definition of the PEG system Σ the $a_1 \dots a_{l_i} \rightarrow \beta_i$ is the i-th rule of the grammar G. Corresponding derivation in the grammar G is:

$$w_{i_1} a_1 a_2 \dots a_{l_i} w_{i_2} \Rightarrow w_{i_1} \beta_i w_{i_2}.$$

Because $\gamma(w_i) = w_{i_1} a_1 a_2 \dots a_{l_i} w_{i_2}$ and $\gamma(w_{i+1}) = w_{i_1} \beta_i w_{i_2}$, hence $L(\Sigma, \omega) \subseteq L(G)$.

Because $L(\Sigma, \omega) \subseteq L(G)$ and $L(G) \subseteq L(\Sigma, \omega)$ hence $L(\Sigma, \omega) = L(G)$.

Hence we showed that the class of the languages defined by the pure grammars of type 0 is subclass of the class of the languages defined by the positioned eco-grammar systems.

According to the [3, p. 202, claim 5.1.5. i)] for the PEG language L from the Example 2 it holds: $L \notin \mathcal{L}(pRE)$. Hence the class of the languages defined by the pure grammars of type 0 is a proper subclass of the class of the languages defined by the positioned eco-grammar systems.

5. Conclusion

The main result of the paper formulated in Section 1 states that family of pure recursively enumerable languages is properly included in the family of *PEG* languages. Together with the results stated in [5] we are able to cover with *PEG* systems all pure (regulated) languages (without appearance checking) languages.

Acknowledgement

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

Research was also supported by the SGS/24/2013 Project of the Silesian University in Opava.

References

[1] Csuhaj-Varjú E., Kelemen J., Kelemenová A., Păun G.: Eco-grammar systems. A grammatical framework pro studying lifelike interactions, Artificial Life, 3, 1997, pp. 1–28.

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

[3] Dassow J., Păun G.: Regulated Rewriting in Formal Language Theory, Akademie – Verlag, Berlin, 1986.

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

[5] Kelemenová A., Langer M.: Positioned Agents in Eco-Grammar Systems, International Journal of Foundations of Computer Science, 22, 2011, pp. 237–246.

[6] Kelemenová A., Langer M.: Positioned agents in eco-grammar systems with border markers and pure regulated grammars, Kybernetika, 48, 2012, pp. 502–517.

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

[8] Langer M.: Agents placed in the environment of eco-grammar systems – Positioned eco-grammar systems, 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.

[9] Langer M.: On Generative Power of Positioned Eco – grammar systems, 1^{st} International Workshop on Formal Models, WFM '06, (Kolar D., Meduna A., eds.), Ostrava, 2006, pp. 35–42.

[10] Martin-Vide C., Păun G.: PM-Colonies, Computers and Artificial Intelligence, 17, 1998, pp. 553–582.

[11] Păun G., Salomaa A.: Families Generated by Grammars and L Systems, Handbook of Formal Languages, Vol. 1. (Rozenberg G., Salomaa A., eds.) Springer, Berlin, 1997, pp. 811–859.