On Generative Power of Positioned Eco-grammar Systems

Miroslav Langer *
miroslav.langer@fpf.slu.cz

Abstract: Positioned eco-grammar systems (PEG systems, for short) were introduced in [5]. In this paper we compare generative power of PEG systems with generative power of PM colonies and Turing machine. We extend results mentioned in [5] and [6]. We also muse on possibilities of extension of generative power of PEG systems.

Keywords: Eco-grammar systems, positioned eco-grammar systems, PM colonies, Turing machine

1 Introduction

Motivation of introducing PEG systems is an attempt to describe interplay between evolving environment, outer influences and community of agents living in this environment whereas we focus on embodiment of the agents and their presence in the environment. PEG systems are based on eco-grammar systems which were introduced in [1] and expanded in [2]. We simply place a new community of agents so called "inner agents" into an environment of eco-grammar system. Agents from original eco-grammar systems are considered as an "outer agents" or "outer influences" which influence the development of the environment. In the case of PEG systems we do not consider the community of "outer agents". Absence of outer agents can be understood as an environment without outer influences e.g. laboratory conditions.

The main difference between PEG systems and eco-grammar systems is in embodiment of the inner agents. Agents of eco-grammar systems can act whenever in environment in fact. In case of PEG systems, this is not possible. The embodiment assures that the agent can change in two following derivation steps symbol distant one symbol at most from the place it acts before. If the reader see some similarity to the PM colonies, he/she is right. The motivation of introducing PEG systems goes from PM colonies too and as we will show in this paper, family of languages defined by PM colonies is a proper subset of the family of languages defined by PEG systems. The inspiration found in PM colonies is presence of agents in the environment, the embodiment of the inner agents.

2 Basic notation

Note 1. We expect that the reader has basic knowledge of formal language theory especially of Lindenmayer systems (see [3], [8]), PM colonies (see [7]) and eco-grammar systems (see [1], [2]).

Let us recall definition of the PM colony and 0L scheme:

Definition 2. PM-colony is a construct $C_{PM} = (V, \#, N, R_1, ...R_n)$, where V is a finite alphabet, $\# \notin V$ is a boundary marker, $N = \{A_1, ..., A_n\}$ is finite alphabet of agents names and $R_1, ..., R_n$ are finite sets of agent's rules over V.

* Institute of Computer Science, Silesian University Opava, Czech Republic

The rules are of the form:

- Deletion:
  - $(a, A_i, b) \rightarrow (\varepsilon, A_i, b)$where $a \in V, b \in V \cup \{\#\}$
  - $(a, A_i, b) \rightarrow (a, A_i, \varepsilon)$ where $a \in V \cup \{\#\}, b \in V$
- Insertion:
  - $(a, A_i, b) \rightarrow (a, c, A_i, b)$ where $a, b \in V \cup \{\#\}, c \in V$
  - $(a, A_i, b) \rightarrow (a, A_i, c, b)$ where $a, b \in V \cup \{\#\}, c \in V$
- Substitution:
  - $(a, A_i, b) \rightarrow (c, A_i, b)$ where $a, c \in V, b \in V \cup \{\#\}$
  - $(a, A_i, b) \rightarrow (a, A_i, c)$ where $a \in V \cup \{\#\}, b, c \in V$
- Move:
  - $(a, A_i, b) \rightarrow (a, b, A_i)$ where $a \in V \cup \{\#\}, b \in V$
  - $(a, A_i, b) \rightarrow (A_i, a, b)$ where $a \in V, b \in V \cup \{\#\}$
- Death:
  - $(a, A_i, b) \rightarrow (a, b)$ where $a, b \in V \cup \{\#\}$

Let $(a, A_i, b) \rightarrow \alpha$ be an action rule of the agent A_i, then the symbols a, b present the context of the agent A_i.

Agent can delete or change (substitute) symbol on its right or left hand side, insert symbol to its right or left hand side, move on one symbol to the left or right, or die - delete itself.

An axiom of the PM colony is a string $\#w\#$ where $w \in (V \cup N)^+$.

Configuration of the PM colony is a string $\#w\#$, where $w \in (V \cup N)^*$.

Derivation step of the PM colony is a binary relation $\Rightarrow$ over a set of configurations of PM colony $\#w\# \Rightarrow \#z\#$ iff each agent A_i which occurs in w replaces it's context in w with right side of proper rule and the resultant string is $\#z\#$. $\Rightarrow^*$ is a reflexive and transitive closure of relation $\Rightarrow$.

Language defined by PM colony is a set of all words derived from the axiom w_0 without agent's names and boundary markers.

$$L(C_{PM}, w_0) = \{x \in E^* : \#w_0\# \Rightarrow \#w\#, x = pr_E(w)\},$$

where $pr_V : (V \cup \{\#\} \cup N)^* \rightarrow V^*$ is a morphism such that $pr_V(a) = a$, for $a \in V$ and $pr_V(b) = \varepsilon$, for $b \in N \cup \{\#\}$.

Family of all languages defined by PM colonies is denoted PM.

Definition 3. 0L scheme is a construct $E = (V_E, P_E)$, where V_E is a finite nonempty alphabet and P_E is a finite set of $0L$ rules over V_E

3 Positioned eco-grammar systems

Positioned eco-grammar systems were introduced in [5]. Let us recall definition of positioned eco-grammar systems.

Definition 4. Positioned eco-grammar system (PEG system, for short) Σ degree m, $m \geq 1$, is $(m+1)$ - tuple $\Sigma = (E, B_1, \ldots, B_m)$, where

- $E = (V_E, P_E)$ is $0L$ scheme - environment
  - V_E - finite nonempty alphabet
  - P_E - finite set of $0L$ rules over V_E
- $B_j = ([j], V_E, Q_j), 1 \leq j \leq n$ is a type of inner agent, where
  - $[j] \in N_B$ is an identifier for presence of $j - th$ agent in environment, where
    - $N_B = \{[j] : 1 \leq j \leq m\}$ is set of identifiers of inner agents

- Q_j - is a set of rules of type $a[j] \rightarrow u$, and $[j]b \rightarrow v$, where $a, b \in V_E$ is symbol marking vicinity with inner agent, $u, v \in (V_E \cup N_B)^*$

Inner agent can arise or die via the rules and its existence in an environment is given by symbol from the set N_B. Agents work parallel. Each agent must in each derivation step rewrite one symbol otherwise the derivation is blocked.

Definition 5. A configuration of the PEG system Σ is a string

$$v = \alpha_0[j_1] \dots \alpha_{m-1}[j_m]\alpha_m,$$

where

$$\alpha_{k-1} \in V_E^*, [j_k] \in N_B, 1 \leq k \leq m, m \geq 0.$$

The starting configuration will be called an axiom.

Definition 6. A derivation step of PEG system Σ is a binary relation $\Rightarrow_\Sigma$ over $(V_E \cup N_B)^*$, such that $w \Rightarrow_\Sigma w'$, iff

- $w = \alpha_0 a_1[j_1]b_1\alpha_1 \dots \alpha_{m-1}a_m[j_m]b_m\alpha_m$, where $\alpha_{k-1} \in V_E^*, a_k, b_k \in (V_E \cup \{\varepsilon\}), [j_k] \in N_B$
- $w' = \alpha_0'\beta_1\alpha_1' \dots \alpha_{m-1}'\beta_m\alpha_m'$

where either $a_k[j_k] \rightarrow \beta_k \in Q_k$ and $a_k \in V_E, b_k = \varepsilon$, or $[j_k]b_k \rightarrow \beta_k \in Q_k$ and $a_k = \varepsilon, b_k \in V_E$ and $\alpha_k \Rightarrow_E \alpha_k'$ for $1 \leq k \leq m$.

Each agent rewrites one symbol on its right or left hand side in each derivation step. The rest of the unchanged symbols are rewritten by the rules of the environment. All agents work parallel. In the case of conflict situation or agents inactivity the derivation is blocked. By the "conflict situation" we understand that two agents want to change the same symbol. By the "inactivity" we understand that agent cannot apply any of it's rule.

Definition 7. Language defined by PEG system Σ and axiom w is a set of strings

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

where $\gamma : (V_E \cup N_B)^* \rightarrow V_E^*$ is a morphism such that $\gamma(a) = a$ for $a \in V_E$ and $\gamma(b) = \varepsilon$ for $b \in N_B$.

Language defined by the positioned eco-grammar system is given by all words ignoring inner agents identifiers, which can be derived from axiom. Family of languages defined by positioned eco-grammar systems (PEG languages) is denoted simply PEG.

In the case of positioned eco-grammar system we consider only evolving environment - 0L system and one community of agents - inner agents. The embodiment of agents causes locality of changes. Each agent can change in one derivation step right one symbol marking its vicinity according to its rules. Rest of the symbols unaffected by agents is changed by the rules of environment.

4 Generative power of PEG systems

Let us recall some already known results on generative power of PEG systems.

Proposition 8. *[5] Family of all finite languages is a proper subset of family of PEG languages* $FIN \subsetneq PEG$.

Proposition 9. *[5] Family of 0L languages is a proper subset of family of PEG languages* $0L \subsetneq PEG$.

Now we will present new results in research of PEG systems.

Theorem 10. *Family of languages generated by PM colonies is a proper subset of family of PEG languages.* $PM \subsetneq PEG$.

The idea of the proof: Consider PM colony C generating language $L(C, w)$. We will construct PEG system Σ such that $L(C, w) = L(\Sigma, w')$. Consider two types of agents. The first type is a deleting agent and the second one is generating agent. To simulate behavior of PM colony agent we need to construct pair of these two types of agents because PM colony agents use twosided context. Our simulation of PM colony by a PEG system is nondeterministic and predictive. We have to guess next move of each PM colony agent and to generate corresponding pair of agents. We also need to keep the information about context. For this purpose we use proper type of agents in our pair. The last thing to solve is a boundary marker. We simply add this symbol into the alphabet of environment and into the alphabet of agents of PEG system. We extend morphism γ of Definition 5 by the rule $\gamma(\#) = \varepsilon$ where $\#$ is a boundary marker.

Proof. Consider PM colony $C = (V, \#, N, R_1, ...R_n)$, where V is finite alphabet, $\# \notin V$ is boundary marker, $N = \{A_1, ..., A_n\}$ is finite alphabet of agents names and $R_1, ..., R_n$ are finite sets of agent's rules over V.

The axiom is a string $\#w\#$ where $w \in (V \cup N)^+$.

We construct PEG system Σ such that $L(C, w) = L(\Sigma, w')$. Let $E = (V_E, P_E)$ where $V_E = V \cup \{\#\}$ is an alphabet and $P_E = \{a \to a : a \in V_E\}$ is a set of rules of 0L scheme of environment. The rules of PM colony have twosided context. This will be simulated by two neighboring agents, one for left context and the other for right context. Consider $B_{0_a} = ([0_a], V_E, \{a[0_a] \to \varepsilon\})$ where $a \in V_E$ first type of agent, deleting agent. Consider $B_{i_a} = ([i_a], V_E, Q_{i_a})$, where $1 \leq i \leq n$ and $a \in V_E$ second type of agent, generating agent. The action rules of generating agent are defined to correspond to the action rules of the PM colony agents and they are as follows:

- Deletion:
  - $[i_a]b \to [0_x][i_x]b$, where $a \in V_E/\{\#\}, b, x \in V_E$ for $(a, A_i, b) \to (\varepsilon, A_i, b) \in R_i$, where $a \in V, b \in V \cup \{\#\}$
  - $[i_a]b \to a[0_a][i_a]$, where $a \in V_E, b \in V_E/\{\#\}$ for $(a, A_i, b) \to (a, A_i, \varepsilon) \in R_i$ where $a \in V \cup \{\#\}, b \in V$
- Insertion:
  - $[i_a]b \to ac[0_c][i_c]b$, where $a, b \in V_E, c \in V_E/\{\#\}$ for $(a, A_i, b) \to (a, c, A_i, b) \in R_i$, where $a, b \in V \cup \{\#\}, c \in V$
  - $[i_a]b \to a[0_a][i_a]cb$ where $a, b \in V_E, c \in V_E/\{\#\}$ for $(a, A_i, b) \to (a, A_i, c, b) \in R_i$, where $a, b \in V \cup \{\#\}, c \in V$
- Substitution:
  - $[i_a]b \to c[0_c][i_c]b$, where $a \in V_E, b, c \in V_E/\{\#\}$ for $(a, A_i, b) \to (c, A_i, b) \in R_i$, where $a \in V, b \in V \cup \{\#\}$
  - $[i_a]b \to a[0_a][i_a]c$, where $a \in V_E, b, c \in V_E/\{\#\}$ for $(a, A_i, b) \to (a, A_i, c) \in R_i$, where $a, b \in V \cup \{\#\}, c \in V$
- Move:
  - $[i_a]b \to ab[0_b][i_b]$, where $a \in V_E, b \in V_E/\{\#\}$ for $(a, A_i, b) \to (a, b, A_i) \in R_i$, where $a \in V \cup \{\#\}, b \in V$
  - $[i_a]b \to [0_x][i_x]ab$, where $a \in V_E/\{\#\}, b, x \in V_E$ for $(a, A_i, b) \to (A_i, a, b) \in R_i$, where $a \in V, b \in V \cup \{\#\}$
- Death:
  - $[i_a]b \to ab$, where $a, b \in V_E$ for $(a, A_i, b) \to (a, b) \in R_i$, where $a, b \in V \cup \{\#\}$

Axiom of the PEG system Σ is a string $w' = \varphi(w)$ where $\varphi : (V \cup N)^* \to ((V_E/\{\#\}) \cup N_B)^*$ is a morphism $\varphi(a) = a$ for $a \in V$ and a is not the context of any agent, $\varphi(aA_ib) = a[0_a][i_a]b$ for $a, b \in V, A_i \in N$. We replace all PM colony agents with corresponding pairs of PEG system agents with respect to the PM colony agent's context. We also need to extend the morphism γ. We add the rule $\gamma(\#) = \varepsilon$. Our PEG system is complete now. Let us show, that this PEG system simulates PM colony C. Each PM colony agent has set of corresponding pairs of agents in PEG system. These agents have the same rules (with respect to the system of course) like the PM colony ones. The context is given by the name of the agent. Only the deleting agent with proper name can delete symbol marking its vicinity than the generating agent in the pair can generate this symbol back, if it is necessary. So if we will generate another pair of agents the derivation will be blocked. In the case of deleting left context symbol or moving to the left we have to guess which symbol is next to the deleted or skipped symbol. If we are wrong the derivation is blocked because we will generate deleting agent which wouldn't be able to delete this symbol. □

We will illustrate the construction on simple example.

Example 1. Consider PM colony $C = (\{a, b\}, \#, \{A_1\}, R_1)$ where
$R_1 = \{\ (a, A, a) \to (A, a, a), (\#, A, a) \to (\#, b, A, a), (b, A, a) \to (b, A, b),$
$(b, A, b) \to (b, b, A),\ \ (b, A, \#) \to (b, \#)\}.$

Let the axiom of the PM colony is a string $w = \#aaAaa\#$.

According to the proof we construct PEG system $\Sigma = (E, B_{0_a}, B_{0_b}, B_{0_\#}, B_{1_a}, B_{1_b}, B_{1_\#})$ where
$E = (\ \{a, b, \#\}, \{a \to a, b \to b, \# \to \#\})$ is 0L scheme of environment,
$B_{0_a} = (\ [0_a], V_E, \{a[0_a] \to \varepsilon\})$,
$B_{0_b} = (\ [0_b], V_E, \{b[0_b] \to \varepsilon\})$,
$B_{0_\#} = (\ [0_\#], V_E, \{\#[0_\#] \to \varepsilon\})$ are deleting agents,
$B_{1_a} = ([1_a], V_E, Q_{1_a})$ *where*
$Q_1 = \{\ [1_a]a \to [0_a][1_a]aa,\ \ [1_a]a \to [0_b][1_b]aa,$
$[1_a]a \to [0_\#][1_\#]aa\}$
$B_{1_b} = ([1_b], V_E, Q_{1_b})$ *where*
$Q_{1_b} = \{[1_b]a \to b[0_b][1_b]b, [1_b]b \to bb[0_b][1_b],$
$[1_b]\# \to b\#\}$
$B_{1_\#} = ([1_\#], V_E, Q_{1_\#})$ *where*
$Q_{1_\#} = \{[1_\#]a \to \#b[0_b][1_b]b\}$ are generating agents.

Axiom of PEG system will be $w' = \#aa[0_a][1_a]aa\#$.
Let us compare derivation in PM colony with derivation in PEG system. PM colony derivation:

$$\begin{aligned}\#aaAaa\# &\Rightarrow \#aAaaa\# \Rightarrow \#Aaaaa\# \Rightarrow \#bAaaaa\# \Rightarrow \#bAbaaa\# \Rightarrow\\ &\Rightarrow \#bbAaaa\# \Rightarrow \#bbAbaa\# \Rightarrow \#bbbAaa\# \Rightarrow \#bbbAba\# \Rightarrow\\ &\Rightarrow \#bbbbAa\# \Rightarrow \#bbbbAb\# \Rightarrow \#bbbbbA\# \Rightarrow \#bbbbb\#.\end{aligned}$$

PEG system derivation:

$$\begin{aligned}\#aa[0_a][1_a]aa\# &\Rightarrow \#a[0_a][1_a]aaa\# \Rightarrow \#[0_\#][1_\#]aaaa\# \Rightarrow \#b[0_b][1_b]aaaa\# \Rightarrow\\ &\Rightarrow \#b[0_b][1_b]baaa\# \Rightarrow \#bb[0_b][1_b]aaa\# \Rightarrow \#bb[0_b][1_b]baa\# \Rightarrow\\ &\Rightarrow \#bbb[0_b][1_b]aa\# \Rightarrow \#bbb[0_b][1_b]ba\# \Rightarrow \#bbbb[0_b][1_b]a\# \Rightarrow\\ &\Rightarrow \#bbbb[0_b][1_b]b\# \Rightarrow \#bbbbb[0_b][1_b]\# \Rightarrow \#bbbbb\#.\end{aligned}$$

This derivation is the only right to simulate our PM colony. If we will chose another rule during moving the agent left, e.g. instead of

$$\#aa[0_a][1_a]aa\# \Rightarrow \#a[0_a][1_a]aaa\# \text{ the } \#aa[0_a][1_a]aa\# \Rightarrow \#a[0_b][1_b]aaa\#$$

the derivation will be blocked in the next step because agent $[0_b]$ has no rule to delete the symbol a on its left side.

Theorem 11. *Family of PEG languages is the proper subset of family of RE languages* $PEG \subsetneq RE$.

Proof. To proof that PEG systems cannot define all RE languages we have to find some language L such that $L \in RE$ and $L \notin PEG$. This will also show some limits of generative power of PEG systems. Agent moving in the environment may suggest head of the Turing machine. This could evoke feeling that we can reach RE generative power. But the main problem is that we do not use the nonterminal symbols, so each string generated during the derivation belongs to the language defined by the PEG system. We will show that for example language $L = \{a^{n^n} : n \geq 0\}$ cannot be generated by PEG system.

Consider PEG system $\Sigma = (E, B_1)$. Without loss of the generality consider only one type of agent $B_1 = ([1], V_E, Q_1)$ (no matter how many types of agents we consider, in the proof we consider the longest string which the agent can derive). Let $E = (V_E, P)$ is 0L scheme of environment. Consider w_0 an axiom of the system Σ.
Without loss of generality consider that in each configuration is each symbol occupied by the agent B_1 (see proposition [5] Family of 0L languages is a subset of PEG languages $0L \subseteq PEG$)
Let p be $p = |\gamma(w_0)|$ the length of the axiom of the system without agents identifiers. Let q be $q = max\{|\gamma(\alpha)| : a[1] \rightarrow \alpha \in Q_1, a \in V_E\}$ length of the longest string without agent's identifiers which agent can derive in one derivation step. Consider w_n an n th configuration.

By the mathematic induction we can proof that $|\gamma(w_n)| \leq p \cdot q^n$.
In the first derivation step $|\gamma(w_1)| \leq p \cdot q$, because from each symbol of axiom we can derive at most q new symbols so the statement holds.
Assume that it also holds for arbitrary n, $|\gamma(w_n)| \leq p \cdot q^n$ and we will show that it also holds for $n + 1$. In the $n + 1$ th step we can derive from each symbol of the configuration at most q new symbols. So we get $|\gamma(w_{n+1})| \leq (p \cdot q^n) \cdot q \leq (p \cdot q^{n+1})$ and our statement holds.

So generally we are able to generate language $L = \{a^{k^n} : n \geq 0, k \in N\}$ for some constant k at most. □

5 Extension of generative power of PEG systems

This chapter is only such a pause on possibility of extension of generative power of PEG systems. Our main goal is not to find the way how to get all RE languages but to study structure, generative power etc. of our model, compare our model with other models. So we will not pronounce any theorem and will not give any exact proof. Let us just tell the idea how to extend generative power. As it was already told, agent moving in the environment may suggest head of the Turing machine. So if we extend our model about nonterminal symbols, we will get RE generative power. Let us show just the idea which makes us sure of this hypothesis. The head of the TM can read right one symbol on the tape at its position, so can the agent in the environment. If we will match each state of TM with different type of agent of PEG system and define proper set of rules for these types of agents according to the rules of TM, we can simulate this TM by PEG system. It's important to note, that we consider special set of nonterminals, dashed terminals. Whenever we generate word from the language $L(TM)$, we just generate special type of agents which will change dashed nonterminals into terminal symbols.

This work has been supported by the Grant Agency of Czech Republic grants No. 201/04/0528 "Výpočetní aspekty emergence - teorie a experimenty" .

Bibliography

1. Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Păun, G.: Eco(grammar) systems - A preview. In: *Cybernetics and Systems '94*. Trappl, R. eds. World Scientific, Singapore, 1994, 941-948.
2. Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Păun, G.: Eco-grammar systems. A grammatical framework for studying lifelike interactions. In: *Artificial Life 3(1)*, 1997, 1-28.
3. Kari, L., Rozenberg, G., Salomaa, A.: L-systems. In: *Handbook of Formal Languages. Vol.1.* Rozenberg, G., Salomaa, A. eds. Springer, Berlin 1997, 253-324.
4. Kelemenová, A.: Eco-grammar systems. In: *Formal Languages and Applications. Studies in Fuzziness and Soft Computing 148*. C. Martín-Vide, V. Mitrana, G. Paun eds. Springer, Berlin, 2004, 311-322.
5. Langer, M.: Agenty umístěné v prostředí ekogramatických systémů - Poziční ekogramatické systémy. In: *Kognice a umělý život V, svazek 2* (J. Kelemen, V. Kvasnička, J. Pospíchal, sest.), Slezská univerzita, Opava, 2005, s. 339-350.
6. Langer, M.: Agents placed in the environment of eco-grammar systems - Positioned eco-grammar systems. In: *Pre-Procs. of the 1st Doctoral Workshop on Mathematical and Engineering Methods in Computer Science* (M. Češka et al., Eds.), FI MU, Brno, 2005, pp. 31-37.
7. Martin-Vide, C., Păun, G.: *PM-Colonies*, Computers and Artificial Intelligence 17, 1998, 553-582.
8. Păun, G., Salomaa, A.: Families Generated by Grammars and L Systems. In: *Handbook of Formal Languages Vol.1.* Rozenberg, G., Salomaa, A. eds. Springer, Berlin, 1997, 811-859.