On Positioned Eco-grammar Systems and Pure Grammars of Type 0

Miroslav Langer[1]

[1] Institute of Computer Science, Faculty of Philosophy and Science, Silesian University in Opava, 74601 Opava, Czech Republic

Malenovice 2012,
$10^{th} - 13^{th}$ of May 2012, Malenovice, Czech Republic

Outline

Eco-grammar systems

- Attempt to describe interaction between evolving environment and the community of agents living within
- Agents – own evolution, act in the environment
- Environment – 0L system
- State of the environment affects the evolution of the agent
- State of the agent affects the evolution of the environment

PM colonies

- Bounded environment - special boundary markers
- Environment – static
- Agents – acts within the environment according to the context
  - delete the symbol
  - insert the symbol
  - substitute the symbol
  - movement
  - death of the agent

Our view

- Attempt to describe the interaction between the evolving environment and the community of the agents living within
- Focus on the embodiment of the agent within the environment
- presence of the agent is given by the special symbol
- We can study local changes - acting of the agent
- Evolution of the agent is suppressed; it can be described by the type of the agent
- Motivation - living tissue attacked by viruses

Positioned eco-grammar systems

- We go out from the eco-grammar systems and PM colonies
- Environment is a 0L scheme
- Presence of the agent is given by the special symbol

Positioned eco-grammar systems

Definition

Positioned eco-grammar systems (PEG system for short) $\Sigma = (V_E, N_B, E, B_1, \ldots, B_m)$ where

- V_E is finite nonempty alphabet of the environment
- $N_B = \{[j] : 1 \leq j \leq m\}$ is the set of identifiers of the agents, $[j]$ sets position of the $j - th$ type of agent within the environment
- $E = (V_E, P_E)$ is $0L$ scheme with alphabet V_E and evolution rules P_E - environment
- $B_j = ([j], Q_j)$, is the $j - th$ type of the agent for $1 \leq j \leq m$ and Q_j is the set of the rules of type $a[j]b \to u$ where $ab \in V_E$ is the symbol on the right or the left side of the agent $[j]$ and $u \in (V_E \cup N_B)^*$
- Axiom $\alpha \in (V_E \cup N_B)^*$

Positioned eco-grammar systems

Definition

- Agent can arise or die via rules
- Presence of the agent within the environment is given by the symbol from the set N_B
- In the environment can be several copies of one type of the agent
- Agents work parallel, untouched symbols are rewritten by the rules of the environment
- Language $L(\Sigma, \alpha) = \{\gamma(u) \in V_E^* : \alpha \Rightarrow_\Sigma^* u, u \in (V_E \cup N_B)^*\}$, where γ is morfism $\gamma(a) = a$ for $a \in V_E$ and $\gamma(b) = \varepsilon$ for $b \in N_B$

Pure Grammars

- Chomski grammars - set of symbols is divided into two disjoint subsets
- Language - contains words composed from the terminal symbols only
- Sentence forms are not included in the language
- In the case of the pure grammars the alphabet is not divided
- Language - all of the words generated during the derivation
- The language contains also the process of its derivation - the strings which corresponds to the sentential forms in Chomsky grammars
- Languages generated by pure grammars are called pure languages or languages of the sentential forms
- Consider pure grammars of type 0

Pure Grammars

Definition

Pure grammars of type 0 is a triple $G = (V, P, S)$ where

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

Previous results concerning pure grammars

Positioned eco-grammar systems

- Pure systems
  - Alphabet is not divided into terminal and nonterminal symbols
- Pure regulated context-free grammars
  - Programmed
  - Matrix
  - Random context
- Without appearance checking

PEG systems and pure grammars of type 0

Need to simulate context rules to obtain the same language

- Need to find sequence of the symbols
- Need to delete this sequence in one derivation step
- Generate the right side of the rule

PEG systems and pure grammars of type 0

How to

- Head of the turing machine
- 5 types of the agents
  - Production choosing agent
  - Left side checking agents
  - Deploying agents
  - Countdown deleting agents
  - Generating agent
- $a_1 a_2 \dots a_n \to w$ is i-th rule from G

PEG systems and pure grammars of type 0

Countdown deleting agents

- Countdown deleting agents B_{D_i}
- $1 \leq i \leq m$, m - length of the longest left side of the rules of G
  - $[D_i]a \to [D_{i-1}]a, a \in V_E, 2 \leq i \leq m$
  - $[D_1]a \to \varepsilon : a \in V_E$

PEG systems and pure grammars of type 0

Production choosing agent

- Production choosing agent B_{PC}
  - Wander in the environment
  - Finds first symbol of the left side of some rule - Left side checking agents

PEG systems and pure grammars of type 0

Left side checking agents

- Left side checking agents
  - $a_1 a_2 \dots a_n$ left side of i-th rule from G
  - $B_{R_i^j}, 1 \leq j \leq n-1$
    - $[R_i^j]a_j \to a_j[R_i^{j+1}]$, $[R_i^j]b \to [PC]b, b \in V_E \setminus a_j$
    - $[R_i^n]a_n \to a_n[B_i^n]$, $[R_i^n]b \to [PC]b, b \in V_E \setminus a_n$

PEG systems and pure grammars of type 0

Deploying agents and Generating agent

- Deploying agents and Generating agent
  - $a_1 a_2 \dots a_n$ left side of i-th rule from G
  - $B_{B_i^j}, 2 \leq j \leq n$
    - $a_j[B_i^j] \to [B_i^{j-1}][D_{j-1}]a_j$
  - $a_1[B_i^1] \to [PC]w$

PEG systems and pure grammars of type 0

How it works

- Production choosing agent finds first symbol of the left side of some rule of G
- Left side checking agents checks if there is whole left side of chosen rule
- Deploying agents deploys Countdown deleting agents
- Countdown deleting agents count down and delete all symbol of the left side of the rule except the first one
- Generating agent uses the first symbol of the left side of the rule to generate the right side and Production choosing agent

We obtain the same language in PEG system

Thanks for your attention :-) Feel free to ask your questions!