THE ALGORITHM FOR REDUCING THE NUMBER OF THE NONTERMINAL SYMBOLS IN THE FE0L SYSTEMS

Ing. Ivana RUDOLFOVÁ, Doctoral Degree Programme (1)
Dept. of Information Systems, FIT, BUT
E-mail: rudolfa@fit.vutbr.cz

Supervised by: Dr. Jaroslav Zendulka

ABSTRACT

This paper introduces FE0L (forbidding context-free L systems) and FET0L (FE0L systems with tables) systems with permitting conditions and the algorithm for reducing the number of the nonterminal symbols in FE0L (FET0L) systems. The productions of FE0L (FET0L) systems (grammars) do not have only the forbidding conditions but also other attached strings, called permitting conditions. The derivation step can proceed in these systems, when the sentential form contains all substrings from the multiset of permitting conditions. This multiset is created in each derivation step according to the productions which are used for this derivation. The algorithm produces a new FE0L (FET0L) grammar with permitting conditions, which generates the same language as the original system.

1 INTRODUCTION

ET0L grammars with forbidding conditions (FET0L grammars) represent an important type of parallel grammars (systems). However, these grammars usually contain many nonterminal symbols. It might be inconvenient for some applications of these grammars. Therefore, this paper presents the algorithm for reducing the number of nonterminal symbols in FET0L systems. The algorithm enables to create the system with seven nonterminal symbols for arbitrary input system. The output system describes the same language as the original system. It is possible to use ET0L systems with only forbidding conditions for the resulting grammar, but the majority of the productions will contain too many forbidding conditions. These forbidding conditions can be replaced with one permitting condition. Therefore, we have designed the FET0L systems with permitting conditions for the output system. The following text is divided into two parts. The first part is devoted to the theory of L systems. In the second part, the algorithm itself is presented.

2 L SYSTEMS

L systems (Lindenmayer systems) are parallel rewriting systems, where all letters in a string are simultaneously rewritten in each derivation step. These systems were introduced by

Aristid Lindenmayer to model the development of filamentous organisms [1], in which cells can communicate and interact with each other. If the rewriting of a letter depends on m of its left and n of its right neighbors, where (m, n) is fixed pair of integers, then the system is system with interactions – an IL system. In this paper, we concentrate on the L systems without interactions (the systems where $m = n = 0$). These systems are denoted as 0L systems. L systems usually do not use any nonterminal symbols, the 0L systems, which use nonterminal symbols, are denoted as E0L systems (extended 0L systems). The productions of E0L system can be divided into several groups and in one derivation step only the productions from the same group can be used. These systems are denoted as ET0L systems (the group of a production is called table). In the following subsections, ET0L grammars, forbidding ET0L grammars and FET0L grammars with permitting conditions are described.

2.1 ET0L GRAMMARS

An ET0L grammar is a $t+3$-*tuple*, $G = (V, T, P_1, ..., P_t, S)$, $t \geq 1$, where *V*, *T* and *S* are the total alphabet, the terminal alphabet $(T \subset V)$, and the axiom $S \in (V - T)^+$, respectively. Each P_i is a finite set of productions of the form $a \to x$, where $a \in V$ and $x \in V^*$. If $a \to x \in P_i$ implies $x \neq \varepsilon$ (ε denotes an empty string) for all $i \in \{1, ..., t\}$, G is said to be *propagating* (EPT0L grammar for short).

Let $u, v \in V^*$, $u = a_1 a_2 ... a_q$, $v = v_1 v_2 ... v_q$, $q = |u|$, $a_j \in V$, $v_j \in V^*$ and $p_1, p_2, ..., p_q$ is a sequence of productions of the form $p_j = a_j \to v_j \in P_i$ for all $j = 1, ..., q$, for some $i \in \{1, ..., t\}$. Then, *u* directly derives *v* according to the productions p_1 through p_q, denoted by $u \Rightarrow_G v \left| p_1, p_2, .., p_q \right.$.

In the standard manner, we define relations $\Rightarrow_G^n$, $\Rightarrow_G^+$ and $\Rightarrow_G^*$ ($n \geq 0$). The language of *G*, denoted by *L(G)*, is defined as $L(G) = \{w \in T^* : S \Rightarrow_G^* w\}$.

Let $G = (V, T, P_1, ..., P_t, S)$ be an ET0L grammar. If $t = 1$, *G* is called an E0L grammar.

2.2 FORBIDDING ET0L GRAMMARS

Forbidding ET0L systems are ET0L systems, whose productions have some attached strings, called forbidding conditions. These systems can make a derivation step only by using productions, whose forbidding conditions do not appear in the rewritten sentential form. The formal description of such systems follows.

A forbidding ET0L grammar (FET0L grammar for short) is defined as a $t+3$-*tuple*, $G = (V, T, P_1, ..., P_t, S)$, $t \geq 1$, where *V*, *T* and *S* have the same meanings as in ET0L grammar, and each P_i is a finite set of productions of the form $(a \to x, F)$, where $a \in V$, $x \in V^*$, and a finite set $F \subseteq V^+$ of forbidding conditions [2].

Let $u, v \in V^*$, $u = a_1 a_2 ... a_q$, $v = v_1 v_2 ... v_q$, $q = |u|$, $a_j \in V$, $v_j \in V^*$ and $p_1, p_2, ..., p_q$ is a sequence of productions $p_j = (a_j \to v_j, F_j) \in P_i$ for all $j = 1, ..., q$ and some $i \in \{1, ..., t\}$, such that $sub(u) \cap \bigcup_{j=1}^{q} F_j = \varnothing$ ($sub(u)$ denotes the set of all nonempty substrings of *u*).

Then, *u* directly derives *v* according to $p_1, p_2,..., p_q$ in *G*, denoted by $u \Rightarrow_G v \left| p_1, p_2,.., p_q \right.$.

The language of *G* is defined as $L(G) = \{x \in T^* : S \Rightarrow_G^* x\}$ [2].

By analogy with ET0L grammars, if $t = 1$, then *G* is called an FE0L grammar.

2.3 FET0L GRAMMARS WITH PERMITTING CONDITIONS

These systems were derived from FE0L systems for the output system of the algorithm, which is presented in this paper. One permitting condition can replace many forbidding conditions in the systems, which are produced by the presented algorithm. Therefore, we have designed these grammars. Productions of these systems may have some attached strings, called permitting conditions. During the derivation step, permitting conditions from all used productions are stored in a multiset of permitting conditions. At the end of the derivation step, it is checked, whether the created string contains such substrings, which are in the multiset of permitting conditions. Here is the formal description of such system:

FET0L system with permitting conditions is a $t+4$-*tuple*, $G = (V, T, P_1,..., P_t, S, Z)$, $t \geq 1$ where *V* is the total alphabet, *T* is the terminal alphabet $(T \subset V)$ and *S* is the axiom $S \in (V - T)^+$. Each P_i is a finite set of productions of the form $(a \rightarrow x, F, D)$, where $a \in V$, $x \in V^*$, and the finite sets $F \subseteq V^+$, $D \subseteq V^+$ (of forbidding and permitting conditions). *Z* is a multiset of permitting conditions.

Let $u, v \in V^*$, $u = a_1 a_2 ... a_q$, $v = v_1 v_2 ... v_q$, $q = |u|$, $a_j \in V$, $v_j \in V^*$ and $p_1, p_2,..., p_q$ is a sequence of productions $p_j = (a_j \rightarrow v_j, F_j, D_j) \in P_i$ for all $j = 1,..., q$ and some $i \in \{1,...,t\}$, such that $sub(u) \cap \bigcup_{j=1}^{q} F_j = \varnothing$, $Z = \bigcup_{j=1}^{q} D_j$. Then, *u* directly derives *v*, if $sub(v) \cap Z = Z$, otherwise *u* is not derived by *G*.

The language of *G* is defined as $L(G) = \{x \in T^* : S \Rightarrow_G^* x\}$.

3 THE ALGORITHM FOR REDUCING THE NUMBER OF NONTERMINAL SYMBOLS IN FE0L SYSTEMS

In this section the algorithm itself is described. The first subsection contains the formal description of the algorithm, the second subsection describes how the new system derivates strings and the last subsection describes the main ideas of the algorithm.

3.1 THE REDUCTION OF FE0L SYSTEMS

- **Input:** FE0L system $G = (V, T, P, s)$ with more than 7 nonterminal symbols.
- **Output:** FE0L system $H = (V', T, P', s')$ with permitting conditions with 7 nonterminal symbols $\{0,1,0',1',2, C, X\}$.
- **Algorithm:**

  1. Define arbitrary injective homomorphism $h: V^* \rightarrow \{0,1\}^*$, such that $h(\varepsilon) = \varepsilon$,

$h(a) \in \{0,1\}^n \{2\}$, $h(a\alpha) = h(a)h(\alpha)$ for all $a \in V$ and $\alpha \in V^*$ and for $n = \lfloor \log(|V|+1) \rfloor$.

2. Define homomorphism $h' : V^* \to \{0',1'\}^*$, which will generate for a specific string the same binary code as homomorphism h, where symbols 0 will be substituted by $0'$ and symbols 1 by $1'$.

3. Set $V' = \{0,1,0',1',2,C,X\} \cup T$ and $s' = h(s)C$.

4. Find the set of productions P', which will contain these productions:

4.1 $(2 \to h'(a)h(\alpha), F \cup \{X\}, \{h'(a)h'(a)\}) \in P'$ for all productions $(a \to \alpha, F) \in P$ (where $a \in V$ and $\alpha \in V^*$)

4.2 $(2 \to h'(a)a, \{C\}, \{h'(a)h'(a)\}) \in P'$ for all $a \in T$

4.3 $(a \to a, \{C\}, \varnothing) \in P'$ for all $a \in T$

4.4 $\left\{ \begin{matrix} (C \to C, \varnothing, \varnothing), (C \to X, \varnothing, \varnothing), (X \to \varepsilon, \varnothing, \varnothing), (0 \to 0', \varnothing, \varnothing), \\ (1 \to 1', \varnothing, \varnothing), (0' \to \varepsilon, \varnothing, \varnothing), (1' \to \varepsilon, \varnothing, \varnothing) \end{matrix} \right\} \subseteq P'$

The algorithm can be also used for ET0L and FET0L systems. In these systems we apply the step 4 of the algorithm on each table of productions of the input system. The new system has the same number of tables as the input system.

3.2 DERIVATION OF STRING IN NEW SYSTEM

When the system *G* generates the string $x \in T^*$ in *n* steps, then system *H* generates the same string in $n+2$ steps. In first *n* steps the system *H* simulates the derivation in the original system by using productions described in 4.1. During this part of derivation, there is the nonterminal *C* at the end of the string. In the last step of this part of derivation, the binary code of string *x* is generated and the nonterminal *C* is rewritten to *X* (in this step we obtain sentential form: $h(x)X$).

In the next step, the terminal symbols are generated from their binary codes by using productions described in 4.2 and nonterminal *X* is rewritten to ε. We obtain a string, where there is its doubled binary code $h'(a)h'(a)$ before each terminal symbol. This step cannot be performed if the string contains binary code of some nonterminal symbol. This symbol can be rewritten with some production described in 4.1, but this production cannot be used because of nonterminal *X* which is contained in the string (*X* is one of the forbidding conditions in productions described in 4.1).

In the last step, all symbols $0'$ and $1'$ are rewritten to ε. In this step the productions described in 4.3 are used. In this step we obtain the string *x*. The derivation can proceed, but we can use only the production described in 4.3, so the result of the following steps will be again the string *x*.

3.3 MAIN IDEAS OF THE ALGORITHM

For all symbols from the input system the binary code is created. The nonterminal symbol *2* is used as a separator of binary codes of symbols (in productions, in axiom and in forbidding conditions). Every binary code of a symbol (of input system) in sentential form is followed by the symbol *2*. Therefore, we can substitute all productions from the input system

with productions with symbol *2* on the left side. (The algorithm is designed for the context-free systems, so there can be only one symbol on the left side of each production – terminal or nonterminal. Productions cannot be substituted with productions for binary codes of symbols, because they contain more than one symbol. For this reason, we use productions with symbol *2* on the left side.)

It is also necessary to ensure, that the production which will be used in the next derivation step corresponds to the production for the symbol which we try to rewrite. Because the new system contains many productions with symbol *2* on the left side, we have to distinguish these productions. We have to ensure the connection between the binary code of a symbol (from the original system) and the production with symbol *2* on the left side, which replaces the production from the input system.

This requirement is reached in the new system by following steps:

- we add productions $0 \to 0'$, $1 \to 1'$, $0' \to \varepsilon$, $1' \to \varepsilon$. When we obtain a binary code $h(a)$ of a symbol a in sentential form, then in the next step, by using these productions, we obtain a binary code $h'(a)$ and in the next step the code is reduced to an empty string
- productions that form the input system are replaced by productions with symbol *2* on the left side and on the right side there is a binary code of the symbol $h'(a)$, which was on the left side of the production in the input system in the form, and then the right side of the original production, where all symbols are replaced with their binary codes
- we add permitting conditions to productions with symbol *2* on the left side. The conditions are of the form $h'(a)h'(a)$, where a is a symbol, which was on the left side of production in the input system

By these steps, we can perform the connection between the symbols from the input system and productions with symbol *2* on the left side. The following example shows how the system works.

4 CONCLUSIONS

The algorithm for reducing the number of the nonterminal symbols in FE0L systems is introduced in this paper. The algorithm produces a new system, which generates the same language as the original system. The output system is FE0L (FET0L) system with permitting conditions, which contains only 7 nonterminal symbols. FE0L (FET0L) systems with permitting conditions are also presented in this paper. We designed these systems especially for our algorithm. The algorithm can be used for FE0L, FET0L, E0L and ET0L systems. The number of forbidding conditions in the input and the output system is the same. The output system is context-free as well as the input system. Only some productions of the output system contain one permitting condition.

REFERENCES

[1] Rozenberg, G., Salomaa, A.: The mathematical theory of L systems, Academic Press, London 1980, ISBN/ISSN 0125971400

[2] Meduna, A., Švec, M.: Forbidding ET0L Grammars, Elsevier Science (2003)