# DELAYED EXECUTION OF SCATTERED CONTEXT GRAMMAR RULES

**Ota Jirák**
Doctoral Degree Programme (2), FIT BUT
E-mail: ijirak@fit.vutbr.cz

Supervised by: Dušan Kolář
E-mail: kolar@fit.vutbr.cz

## ABSTRACT

This paper presents an idea of a delayed parsing of scattered context grammars. We introduce a delayed execution of scattered context grammar rules. This technique uses a method usually used for the top-down analysis of context-free grammars. We extended the usage of LL parsing tables and pushdown automata.

## 1 INTRODUCTION

The family of languages described by scattered context grammars is very important due to its generative power. However, there is no known efficient method for parsing these grammars. That is the motivation to study parsing methods of these languages.

Rules of scattered context grammars are made from rules of context-free grammars. The main idea of this article is to use a context-free parsing method to parse scattered context grammars. The first part contains basic definitions. In the next part, we describe a parsing algorithm. There is description of delayed execution of a scattered context grammar rules. The research of an algorithm selecting the rules is still in progress. Therefore, there is only a suggestion and a basic idea of the rule selection.

## 2 BASIC DEFINITIONS

We assume that the reader is familiar with the theory of formal languages (see [1]).

Some definitions of scattered context and context-free grammars should be presented for illustration and a better understanding. More about this topic is in [2] and [3].

**Definition 2.1 (Context Free Grammar)** *A context free grammar (CFG, for short) is a quadruple $G = (V,T,P,S)$, where $V$ is a finite set of symbols, $T \subset V$ is a terminal alphabet, $S \in V \setminus T$ is the starting nonterminal, and $P$ is a finite set of rules of the form $A \rightarrow w$, where $A \in V \setminus T$ and $w \in V^*$.*

**Definition 2.2 (Scattered Context Grammar)** *A scattered context grammar (SCG, for short) is a quadruple $G = (V,T,P,S)$, where $V$ is a finite set of symbols, $T \subset V$ is a terminal alphabet,*

*$S \in V \setminus T$ is the starting nonterminal, and $P$ is a finite set of rules of the form $(A_1, A_2, \ldots, A_n) \to (w_1, w_2, \ldots, w_n)$, for some $n \geq 1$, where $A_i \in V \setminus T$ and $w_i \in V^*$. If $(A_1, A_2, \ldots, A_n) \to (x_1, x_2, \ldots, x_n) \in P, u = u_1 A_1 u_2 \ldots u_n A_n u_{n+1}$, and $v = u_1 x_1 u_2 \ldots u_n x_n u_{n+1}$, where $u_i \in V$ for $1 \leq i \leq n$, then $u \Rightarrow v$.*

**Definition 2.3 (Indexing of SCG rules)** *Let $p_1, p_2, \ldots, p_n$ be a CFG rules from SCG rule $p = (p_1, p_2, \ldots, p_n)$, where $n \in \mathbb{N}$. For $1 \leq i \leq n$ it holds $p[i] = p_i$. Accordingly for $1 \leq j \leq k \leq n$, we define $p[j:k] = (p[j], p[j+1], \ldots, p[k])$. For $i > n$ it holds $p[i] = \emptyset$.*

**Definition 2.4 (Indexing of SCG rules set)** *Let $P$ be a finite set of SCG rules. Then $P[i]$ is an multiset define as follows: $P[i] = \{p[i] : p \in P, i \in \mathbb{N}\}$.*

## 3 LL SCG GRAMMARS

LL SCGs are often used, because we can use a deterministic algorithms to analyze these grammars.

To make things clear we remind the definition of LL CFG.

**Definition 3.1 (Empty set)** *Let $G = (N, T, P, S)$ be a CFG grammar. $Empty(x) = \varepsilon$ if $x \Rightarrow^* \varepsilon$ else $Empty(x) = \emptyset$, where $x \in (N \cup T)^*$.*

**Definition 3.2 (First set)** *Let $G = (N, T, P, S)$ be a CFG grammar. For each $x \in (N \cup T)^*$ we define $First(x) = \{a \in T : x \Rightarrow^* ay, y \in (N \cup T)^*\}$.*

**Definition 3.3 (Follow set)** *Let $G = (N, T, P, S)$ be a CFG grammar. For each $A \in N$, we define $Follow(A) = \{a : a \in T, S \Rightarrow^* xAay, x, y \in (N \cup T)^* \cup \{\$ : S \Rightarrow^* xA, x \in (N \cup T)^*\}$.*

**Definition 3.4 (Predict set)** *Let $G = (N, T, P, S)$ be a CFG. For each rule $A \to x \in P$, we define $Predict(A \to x)$ as follows:*

- *if $Empty(x) = \varepsilon$ then $Predict(A \to x) = First(x) \cup Follow(A)$,*
- *if $Empty(x) = \emptyset$ then $Predict(A \to x) = First(x)$.*

**Definition 3.5 (LL CFG)** *Let $G = (N, T, P, S)$ be a CFG. $G$ is LL-grammar if for all $a \in T$ and $A \in N$ there is maximal one rule $p = A \to X_1 \ldots X_n \in P$ satisfying the condition $a \in Predict(A \to X_1 \ldots X_n)$.*

Now we define LL Scattered Context Grammars.

**Definition 3.6 (LL SCG)** *Let $G_1 = (N, T, P, S)$ be a SCG. $G_1$ is LL SCG if $G_2 = (N, T, P[1], S)$ is LL CFG.*

## 4 EXISTING PARSING METHOD FOR SCG

For parsing scattered context grammars, we use a deterministic version of Regulated Pushdown Automata (RPDA , see [4]). Syntax analysis using RPDA is very similar to Pushdown Automata (PDA). The basic principle remains the same. An RPDA uses symbol from input string and symbol from the top of the stack to choose the rule. These symbols are index into the LL parse table. Reduction is applied the same way as for a PDA. Expansion is more complicated, there is usually more than one symbol to expand. An RPDA takes out symbols from the stack looking for all nonterminals that must be expanded. During this operation a word from regulation language is created. This word is used for stack reconstruction. Bad time complexity is against efficient usage of this method. More about SCG parsing is in [5].

## 5 DELAYED EXECUTION OF SYNTAX RULES

Parsing method "delayed execution of syntax rules" is based on principles from parsing of LL CFG grammars.

As we can see in definitions 2.1 and 2.2, CFG and SCG are very similar. Rules of scattered context grammars are created from context-free rules. This fact gives rise to the idea of using CFG parsing methods. We extend using of LL table and pushdown automaton (PDA).

We use modified pushdown automaton (MPDA). The basic function of an MPDA works the same as a PDA. Changes are in the rule selection and in the rule processing. MPDAs work with LL parsing table and set of delayed rules (SDR). The SDR is initially empty. The LL parsing table for SCG is constructed in the same way like LL parsing table for CFG. We use only the first CFG part from each SCG rule.

The following algorithm explains the syntax analysis of an LL SCG based on leftmost derivations.

1. Read symbol from the input tape.
2. Use the symbol read for a rule selection. If SDR is not empty, we try to choose delayed rule from SDR. If the selection fails or SDR is empty, we use LL table to select the rule. Research of the key problem, algorithm for rules selection, is still in progress. Therefore, no accurate algorithm is presented.
3. Now we apply first part of given product $p$, $p[1]$ – first CFG rule. There are the same procedures like in PDA (expansion and reduction of the pushdown).
4. Unprocessed part of the product, it is $p[2:n]$, is stored in SDR. If $p$ is delayed rule, we replace it. Such a way, we obtain a set of delayed rules.
5. We repeat the whole procedure until there is no applicable rule or the input string is empty. Conditions for acceptance of string with this automaton are:
   - empty pushdown (can be changed to acceptance with finish state),
   - empty SDR (nonempty means non-processed rules).

To have an unambiguous rule selection, we have to satisfy LL condition for all rules $p \in P[1]$. It means, the analyzed grammar has to be LL SCG. Only the first parts of SCG rules have to satisfy

LL(1) condition. The others need not satisfy this condition. Their occurrence is determined by the first rule.

## 6 EXAMPLE

The best way how to explain of processing of delayed rules is to show it on the following example. We have an input string "*aebfcgdh*", a scattered context grammar $G = (N,T,P,S)$, $N = \{S,A,B,C,D\}$, $T = \{a,b,c,d,e,f,g,h\}$ and

$P = \{$

$$p1 : (S) \rightarrow (ABCD),$$
$$p2 : (A,B,C,D) \rightarrow (aA,bB,cC,dD),$$
$$p3 : (A,B,C,D) \rightarrow (e,f,g,h),$$
$$p4 : (A,B,C,D) \rightarrow (i,f,g,h)$$

$\}$

In Figure 1, we can see the obtained LL table. Processing of the input string is shown in Figure 2.

| | a | b | c | d | e | f | g | h | i | $ |
|---|---|---|---|---|---|---|---|---|---|---|
| S | 1 | | | | 1 | | | | 1 | |
| A | 2 | | | | 3 | | | | 4 | |
| B | | | | | | | | | | |
| C | | | | | | | | | | |
| D | | | | | | | | | | |
| a | | | | | | | | | | |
| b | | | | | | | | | | |
| c | | | | | | | | | | |
| d | | | | | | | | | | |
| e | | | | | | | | | | |
| f | | | | | | | | | | |
| g | | | | | | | | | | |
| h | | | | | | | | | | |
| i | | | | | | | | | | |
| # | | | | | | | | | | |

**Figure 1:** LL table

## 7 CONCLUSION AND FUTURE WORK

The introduced method "delayed execution of scattered context rules" can decrease the time complexity of the parsing of scattered context grammars because we need not use slow time inefficient regulated pushdown automata. Faster compiler will enable us to perform a more efficient data analysis, e.g. virus detection. We will be able to describe suspicious code with

| Pushdown | | | | | | state | input tape | | | | | | | | | action | SDR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | # | S | s | a | e | b | f | c | g | d | h | $ | p1 | |
| | # | D | C | B | A | s | a | e | b | f | c | g | d | h | $ | p2[1] | p2[2:4] |
| # | D | C | B | A | a | s | a | e | b | f | c | g | d | h | $ | red | p2[2:4] |
| | # | D | C | B | A | s | e | b | f | c | g | d | h | $ | | p3[1] | p2[2:4],p3[2:4] |
| | # | D | C | B | e | s | e | b | f | c | g | d | h | $ | | red | p2[2:4],p3[2:4] |
| | | # | D | C | B | s | b | f | c | g | d | h | $ | | | p2[2] | p2[3:4],p3[2:4] |
| | # | D | C | B | b | s | b | f | c | g | d | h | $ | | | red | p2[3:4],p3[2:4] |
| | | # | D | C | B | s | f | c | g | d | h | $ | | | | p3[2] | p2[3:4],p3[3:4] |
| | | # | D | C | f | s | f | c | g | d | h | $ | | | | red | p2[3:4],p3[3:4] |
| | | | # | D | C | s | c | g | d | h | $ | | | | | p2[3] | p2[4],p3[3:4] |
| | | # | D | C | c | s | c | g | d | h | $ | | | | | red | p2[4],p3[3:4] |
| | | | # | D | C | s | g | d | h | $ | | | | | | p3[3] | p2[4],p3[4] |
| | | | # | D | g | s | g | d | h | $ | | | | | | red | p2[4],p3[4] |
| | | | | # | D | s | d | h | $ | | | | | | | p2[4] | p3[4] |
| | | | # | D | d | s | d | h | $ | | | | | | | red | p3[4] |
| | | | | # | D | s | h | $ | | | | | | | | p3[4] | |
| | | | | # | h | s | h | $ | | | | | | | | red | |
| | | | | | # | s | $ | | | | | | | | | acc | |

red - reduce acc - accepted

**Figure 2:** Analysis using delayed execution of rules

scattered context grammars. This type of virus description could be a headstone of a new generation of antiviruses.

The future work will focus on the efficient algorithm for the selection of SCG rules.

*This paper was supported by the Research Plan No. MSM 0021630528 - Security Oriented Research in Information Technology.*

## REFERENCES

[1] Meduna, A.: Automata and Languages: Theory and Applications. Springer-Verlag, London 2000.

[2] Greibach, S. A.; Hopcroft, J. E.: Scattered Context Grammars. *J. Comput. Syst. Sci.*, Vol. 3, No. 3, 1969, p. 233-247.

[3] Masopust, T.: Scattered Context Grammars Can Generate the Powers of 2, In: *Proceedings of the 13th Conference STUDENT EEICT 2007*, Volume 4, Brno, CZ, FEKT VUT, 2007, p. 401- 404, ISBN 978-80214-3410-3.

[4] Kolář, D., Meduna, A.: Regulated Pushdown Automata. Acta Cybernetica, Vol. 2000, No. 4, 2000, p. 653-664, ISSN 0324-721X.

[5] Kolář, D.: Scattered Context Grammars Parsers. In: *Proceedings of the 14th International Congress of Cybernetics and Systems* of WOCS, Wroclaw University of Technology, 2008, ISBN 978-83-7493-400-8, p. 491-500.