Parsing Based on Several Grammars

Is parsing a solo act or a team sport?

Motivation & Problem

In practical compiler construction, different syntactic constructs of real-world programming languages are naturally suited to different parsing methods. Expressions are most elegantly analyzed by operator-precedence or LR methods, while statements and control structures fit LL grammars. However, there is no unified approach for combining multiple parsing methods into a single parser. One solution is sequentially working grammar systems, i.e., CD (Cooperating Distributed) grammar systems, which separate the language into modular components that are active one at a time. A key problem remains: the parser must know when and for how long each component should analyze the input. This is addressed by a communication table that controls switching between components via designated symbols, forming a CTCD (Communication Table-controlled CD) grammar system.

Algorithm 5.1 Evaluation function $\alpha(A, t)$

Input: $G_1, \ldots, G_n$
Output: $\alpha(A, t)$

$M := \emptyset$
$G_{all} := G_1 \cup \cdots \cup G_n$
for all $A \to x \in G_{all}$ **do**
  for all $B \to y \in G_{all}$ **do**
    if $isSubstr(A, y)$ **and** $component(A \to x) \neq component(B \to y)$ **then**
      $M := M \cup \{A\}$
    end if
  end for
end for
$\alpha := \emptyset$
for all $A \to x \in G_{all}$ **do**
  if $A \in M$ **then**
    $Predict(A \to x) := createPredict(A \to x)$
    $G_i := component(A \to x)$
    for all $t \in Predict(A \to x)$ **do**
      $\alpha := \alpha \cup \{((A, t), G_i)\}$
    end for
  end if
end for

Algorithm 1: Evaluation function α(A,t)

System Design

	fun	INT, FLOAT, BOOL, STRING	ID_S	ID_V	)	ID_F	switch	const	INT_LIT,null, FLOAT_LIT,!, STRING_LIT,(, true,false
<FUN_DEF>	G_2								
<TYPE>		G_2	G_2						
<PARAMS>				G_3	G_3				
<FUN_CALL>						G_4			
<SWITCH>							G_5		
<STRUCT_DEF>								G_6	
<STRUCT_INIT>			G_7						
<EXPR>				G_8		G_8			G_8

Table 1: Communication table

$G_{1,\ldots,7}$ – LL / LR

G_8 – Precedence / LR

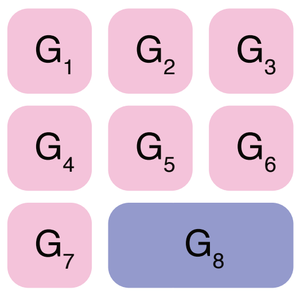

Figure 1: Eight-Component Grammar System

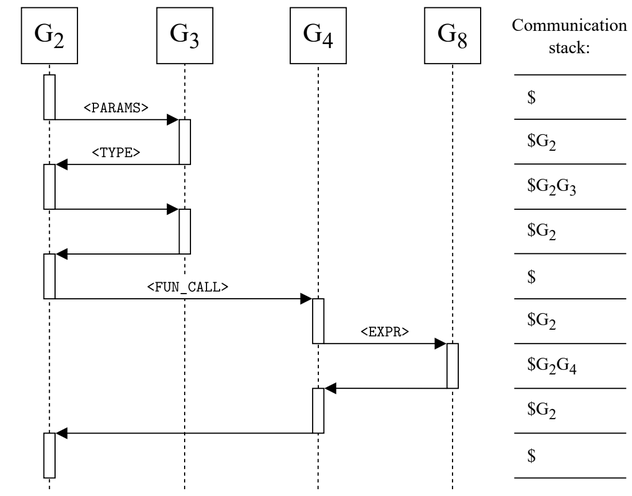

Figure 2: Sequence Diagram

Parser Implementation

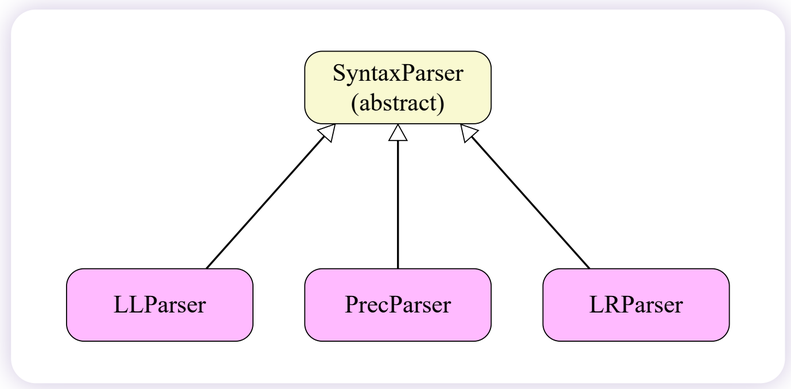

Figure 3: Class Diagram

Abstract Syntax Tree

Figure 4: Abstract Syntax Tree

Author: Daniel Ulvr
Supervisor: prof. RNDr. Alexandr Meduna, CSc.

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2026