T-603-THYD Compilers

CFG for Decaf Suitable for LL(1) parsing

Edda Pétursdóttir Edda Steinunn Rúnarsdóttir

October 5th 2018

This report shows a context-free grammar describing the programming language Decaf that has been rewritten from its original form provided by teacher. The context-free grammar is now suitable for LL(1) top-down parsing. That is, the grammar is rid of left-recursion, has been left-factored and correct associatively and precedence of operators have been ensured (assuming operator associativity and precedence is the same as in the Java programming language).

CFG for Decaf Suitable for LL(1) Parsing

The following is a context-free grammar for Decaf suitable for LL(1) parsing. Note that non-terminals are shown in *italics* and tokens are shown in **bold**

program ::= **class** **id** **{** *variable_declarations* *method_declarations* **}**

variable_declarations ::= *type* *variable_list* **;** *variable_declarations* | ε

type ::= **int** | **real** | **bool**

variable_list ::= *variable* *more_variable_list*

more_variable_list ::= **,** *variable* *more_variable_list* | ε

variable ::= **id** *optional_array*

optional_array ::= **[int_value]** | ε

method_declarations ::= *method_declaration* *more_methods*

more_methods ::= *method_declaration* *more_methods* | ε

method_declaration ::= **static** *method_return_type* **id** **(** *parameters* **)**
{ *variable_declarations* *statement_list* **}**

method_return_type ::= *type* | **void**

parameters ::= *parameter_list* | ε

parameter_list ::= *type* **id** *more_parameter_list*

more_parameter_list ::= **,** *type* **id** *more_parameter_list* | ε

statement_list ::= *statement* *statement_list* | ε

statement ::= **id** *id_statement* **;**
| **if** **(** *expression* **)** *statement_block* *optional_else*
| **for** **(** *variable* **=** *expression* **;** *expression* **;** *incr_decr_var* **)**
statement_block
| **return** *optional_expression* **;**
| **break ;**
| **continue ;**
| *statement_block*

id_statement ::= *optional_array* *assign_or_increment* | **(** *expression_list* **)**

assign_or_increment ::= **=** *expression* | *increment_decrement_op*

optional_expression ::= *expression* | ε

statement_block ::= **{** *statement_list* **}**

incr_decr_var ::= *variable* *incr_decr_op*

incr_decr_op ::= **++** | **−−**

optional_else ::= **else** *statement_block* | ε

expression_list ::= *expression* *more_expressions* | ε

more_expressions ::= **,** *expression* *more_expressions* | ε

value ::= **int_value** | **real_value** | **bool_value**

(CFG for Decaf Continued on Next Page)

(CFG for Decaf Continued)

$$
\begin{array}{rcl}
\mathit{expression} & ::= & \mathit{and\_expression} \quad \mathit{expression}' \\
\mathit{expression}' & ::= & || \quad \mathit{and\_expression} \quad \mathit{expression}' \quad | \quad \varepsilon \\
\mathit{and\_expression} & ::= & \mathit{equal\_expression} \quad \mathit{and\_expression}' \\
\mathit{and\_expression}' & ::= & \mathbf{\&\&} \quad \mathit{equal\_expression} \quad \mathit{and\_expression}' \quad | \quad \varepsilon \\
\mathit{equal\_expression} & ::= & \mathit{comp\_expression} \quad \mathit{equal\_expression}' \\
\mathit{equal\_expression}' & ::= & \mathit{equal\_op} \quad \mathit{comp\_expression} \quad \mathit{equal\_expression}' \quad | \quad \varepsilon \\
\mathit{comp\_expression} & ::= & \mathit{add\_expression} \quad \mathit{comp\_expression}' \\
\mathit{comp\_expression}' & ::= & \mathit{comp\_op} \quad \mathit{add\_expression} \quad \mathit{comp\_expression}' \quad | \quad \varepsilon \\
\mathit{add\_expression} & ::= & \mathit{mul\_expression} \quad \mathit{add\_expression}' \\
\mathit{add\_expression}' & ::= & \mathit{add\_op} \quad \mathit{mul\_expression} \quad \mathit{add\_expression}' \quad | \quad \varepsilon \\
\mathit{mul\_expression} & ::= & \mathit{unary\_expression} \quad \mathit{mul\_expression}' \\
\mathit{equal\_expression}' & ::= & \mathit{mul\_op} \quad \mathit{unary\_expression} \quad \mathit{mul\_expression}' \quad | \quad \varepsilon \\
\mathit{unary\_expression} & ::= & \mathit{unary\_op} \quad \mathit{unary\_expression} \quad | \quad \mathit{primary\_expression} \\
\mathit{mul\_op} & ::= & \mathbf{*} \quad | \quad \mathbf{/} \quad | \quad \mathbf{\%} \\
\mathit{equal\_op} & ::= & \mathbf{==} \quad | \quad \mathbf{!=} \\
\mathit{comp\_op} & ::= & \mathbf{<} \, \mathit{or\_equal} \quad | \quad \mathbf{>} \, \mathit{or\_equal} \\
\mathit{unary\_op} & ::= & \mathbf{!} \quad | \quad \mathbf{+} \quad | \quad \mathbf{-} \\
\mathit{or\_equal} & ::= & \mathbf{=} \quad | \quad \varepsilon \\
\mathit{primary\_expression} & ::= & \mathbf{(} \quad \mathit{expression} \quad \mathbf{)} \\
 & & | \quad \mathit{value} \\
 & & | \quad \mathbf{id} \quad \mathit{id\_addition} \\
\mathit{id\_addition} & ::= & \mathit{optional\_array\_variable} \quad | \quad \mathbf{(} \quad \mathit{expression\_list} \quad \mathbf{)}
\end{array}
$$