Design Decisions and Evaluation of Parser

Maryam Dabaghchian-u1078006

February 6, 2018

1 Operations' Priority and Left Association

Operations' priority and left association is related to Exp production rules in the grammar. So, I change them as follows to follow the correct priority and left association.
$Exp \rightarrow LtLevel$
$Exp \rightarrow Exp\ "\&\&"\ LtLevel$
$LtLevel \rightarrow PlusLevel$
$LtLevel \rightarrow LtLevel\ " < "\ PlusLevel$
$PlusLevel \rightarrow MultLevel$
$PlusLevel \rightarrow PlusLevel\ ("+"|"-")\ MultLevel$
$MultLevel \rightarrow DotLevel$
$MultLevel \rightarrow MultLevel\ " * "\ DotLevel$
$DotLevel \rightarrow PrimaryExp\ (\ "["\ Exp\ "]"\ |\ "."\ "length"\ |\ "."\ Id\ "("\ (\ Exp\ (\ ","\ Exp\ )^*\ )?\ ")"\ )^*$
$DotLevel \rightarrow "!"\ DotLevel$
$PrimaryExp \rightarrow "true"\ |\ "false"\ |\ "this"\ |\ "new"\ "int"\ "["\ Exp\ "]"\ |\ "new"\ Id\ "("\ ")"\ |\ <$
$NUM >\ |\ "("\ Exp\ ")"$

2 Eliminate Left Recursion

Then, in the next step I eliminate the left recursion that exists in production rules defined in Section 1. Other production rules do not have left recursion.
$Goal \rightarrow MainClass\ (\ ClassDecl\ )^*\ < EOF >$
$MainClass \rightarrow "class"\ Id\ "\{"\ "public"\ "static"\ "void"\ "main"\ "("\ "String"\ "["\ "]"\ Id\ ")"\ "\{"\ Stmt\ "\}"\ "\}"$
$ClassDecl \rightarrow "class"\ Id\ (\ "extends"\ Id\ )?\ "\{"\ (\ VarDecl\ )^*\ (\ MethodDecl\ )^*\ "\}"$
$VarDecl \rightarrow Type\ Id\ ";"$
$MethodDecl \rightarrow "public"\ Type\ Id\ "("\ (\ Type\ Id\ (\ ","\ Type\ Id\ )^*\ )?\ ")"\ "\{"\ (\ VarDecl\ )^*\ (\ Stmt\ )^*\ "return"\ Exp\ ";"\ "\}"$
$- -$
$Type \rightarrow "int"\ "["\ "]"\ |\ "boolean"\ |\ "int"\ |\ Id$
$- -$
$Stmt \rightarrow "\{"\ (\ Stmt\ )^*\ "\}"\ |$
$"if"\ "("\ Exp\ ")"\ Stmt\ "else"\ Stmt\ |$
$"while"\ "("\ Exp\ ")"\ Stmt\ |$
$"System.out.println"\ "("\ Exp\ ")"\ ";"\ |$
$Id\ " = "\ Exp\ ";"\ |$

$Id\ "["\ Exp\ "]"\ "="\ Exp\ ";"\ \mid$

$- -$

$Exp \rightarrow LtLevel\ Exp'$
$Exp' \rightarrow "\&\&"\ LtLevel\ Exp'$
$Exp' \rightarrow$
$LtLevel \rightarrow PlusLevel\ LtLevel'$
$LtLevel' \rightarrow "<"\ PlusLevel\ LtLevel'$
$LtLevel' \rightarrow$
$PlusLevel \rightarrow MultLevel\ PlusLevel'$
$PlusLevel' \rightarrow ("+" \mid "-")\ MultLevel\ PlusLevel'$
$PlusLevel' \rightarrow$
$MultLevel \rightarrow DotLevel\ MultLevel'$
$MultLevel' \rightarrow "*"\ DotLevel\ MultLevel'$
$MultLevel' \rightarrow$
$DotLevel \rightarrow PrimaryExp\ (\ "["\ Exp\ "]" \mid "."\ "length" \mid "."\ Id\ "("\ (\ Exp\ (\ ","\ Exp\ )^*\ )?\ ")"\ )^*$
$DotLevel \rightarrow "!"\ DotLevel$
$PrimaryExp \rightarrow "true" \mid "false" \mid "this" \mid "new"\ "int"\ "["\ Exp\ "]" \mid "new"\ Id\ "("\ ")" \mid <NUM> \mid <ID> \mid "("\ Exp\ ")"$

3 Left Factoring

After eliminating the left recursion, I do the left factoring, which enables the parser to decide which production rule should be used next by looking at the next token.

$Goal \rightarrow "class"\ Id\ MainClass\ (\ "class"\ Id\ RegClass\ )^*\ <EOF>$
$MainClass \rightarrow "\{"\ "public"\ "static"\ "void"\ "main"\ "("\ "String"\ "["\ "]"\ Id\ ")"\ "\{"\ Stmt\ "\}"\ "\}"$
$RegClass \rightarrow (\ "extends"\ Id\ )?\ "\{"\ (\ VarDecl\ )^*\ (\ MethodDecl\ )^*\ "\}"$
$VarDecl \rightarrow Type\ Id\ ";"$
$MethodDecl \rightarrow "public"\ Type\ Id\ "("\ (\ Type\ Id\ (\ ","\ Type\ Id\ )^*\ )?\ ")"\ "\{"\ (\ VarDecl\ )^*\ (\ Stmt\ )^*\ "return"\ Exp\ ";"\ "\}"$

$- -$

$Type \rightarrow "int"\ "["\ "]" \mid "boolean" \mid "int" \mid Id$

$- -$

$Stmt \rightarrow "\{"\ (\ Stmt\ )^*\ "\}" \mid$
$"if"\ "("\ Exp\ ")"\ Stmt\ ElseStmt \mid$
$"while"\ "("\ Exp\ ")"\ Stmt \mid$
$"System.out.println"\ "("\ Exp\ ")"\ ";" \mid$
$Id\ AssignStmt$
$ElseStmt \rightarrow (\ "else"\ Stmt\ )?$
$AssignStmt \rightarrow "="\ Exp\ ";" \mid$
$"["\ Exp\ "]"\ "="\ Exp\ ";"$

$- -$

$Exp \rightarrow LtLevel\ Exp'$
$Exp' \rightarrow "\&\&"\ LtLevel\ Exp'$
$Exp' \rightarrow$
$LtLevel \rightarrow PlusLevel\ LtLevel'$
$LtLevel' \rightarrow "<"\ PlusLevel\ LtLevel'$
$LtLevel' \rightarrow$

$PlusLevel \rightarrow MultLevel\ PlusLevel'$
$PlusLevel' \rightarrow ("+"|"-")\ MultLevel\ PlusLevel'$
$PlusLevel' \rightarrow$
$MultLevel \rightarrow DotLevel\ MultLevel'$
$MultLevel' \rightarrow "*"\ DotLevel\ MultLevel'$
$MultLevel' \rightarrow$
$DotLevel \rightarrow PrimaryExp\ (\ "["\ Exp\ "]"\ |\ "."\ "length"\ |\ "."\ Id\ "("\ (\ Exp\ (\ ","\ Exp\ )^*\ )?\ ")"\ )^*$
$DotLevel \rightarrow "!"\ DotLevel$
$PrimaryExp \rightarrow "true"\ |\ "false"\ |\ "this"\ |\ "new"\ NewExp\ |\ <NUM>\ |\ <ID>$
$|\ "("\ Exp\ ")"$
$NewExp \rightarrow "int"\ "["\ Exp\ "]"\ |\ Id\ "("\ ")"$