First and Follow Sets for LR(1) Parser

Compiler Theory - Programming Project 4 : LR(1) Parser
21400646 Lim Chae Eon

Grammar Rule Choices from BNF

Program	→	**<script_start>** *stmt-sequence* **<script_end>**
stmt-sequence	→	*statement* **;** *stmt-sequence*
stmt-sequence	→	*comment* **nextLine**
stmt-sequence	→	ε
statement	→	*if-stmt*
statement	→	*declaration-stmt*
statement	→	*loop-stmt*
statement	→	*assign-stmt*
statement	→	*function-stmt*
statement	→	*switch-stmt*
statement	→	*increment-stmt*
statement	→	**break**
simple-stmt	→	*stmt-sequence*
simple-stmt	→	**{** *stmt-sequence* **}**
if-stmt	→	**if (** *exp* **)** *simple-stmt*
if-stmt	→	**if (** *exp* **)** *simple-stmt* **else** *simple-stmt*
declaration-stmt	→	**var** *id*
id	→	*assign-stmt*
id	→	*id* **,** *id*
id	→	**IDENTIFIER**
assign-stmt	→	**IDENTIFIER** *assign-op exp*
assign-op	→	**=**
assign-op	→	−=
assign-op	→	**+=**
exp	→	*simple-exp logic-op simple-exp*
exp	→	*simple-exp*
logic-op	→	**<**
logic-op	→	**>**
logic-op	→	**<=**
logic-op	→	**>=**
logic-op	→	**==**
logic-op	→	**!=**
simple-exp	→	*simple-exp add-op term*
simple-exp	→	*term*
add-op	→	**+**
add-op	→	−
term	→	*term mul-op factor*

term	→	*factor*
mul-op	→	*****
mul-op	→	**/**
factor	→	**(** *exp* **)**
factor	→	**NUMBER**
factor	→	**IDENTIFIER**
loop-stmt	→	**for (** *for-parameter* **)** *simple-stmt*
loop-stmt	→	**while (** *exp* **)** *simple-stmt*
for-parameter	→	exp **;** exp **;** exp
function-stmt	→	*function-keyword* **(** *function-parameter* **)**
function-keyword	→	**window.prompt**
function-keyword	→	**window**
function-keyword	→	**parseFloat**
function-keyword	→	**document.writeln**
function-keyword	→	**document.write**
function-keyword	→	**document**
function-parameter	→	**IDENTIFIER**
function-parameter	→	**LITERAL**
function-parameter	→	**NUMBER**
increment-stmt	→	*increment-op* **IDENTIFIER**
increment-stmt	→	**IDENTIFIER** *increment-op*
increment-op	→	**++**
increment-op	→	−−
switch-stmt	→	**switch (IDENTIFIER) {** *case-part default-block* **}**
case-part	→	*case-block case-part*
case-part	→	ε
case-block	→	*case-condition* **:** *stmt-sequence*
case-condition	→	**case** *case-parameter*
case-parameter	→	**NUMBER**
case-parameter	→	**LITERAL**
default-block	→	**default :** *stmt-sequence*
default-block	→	ε
comment	→	**// anything**

First Sets

First(Program) = {<script_start>}

First(stmt-sequence) = {ε, break, if, var, for, while, IDENTIFIER, window.prompt, window, parseFloat, document.writeln, document.write, document, switch, //, ++, --}

First(statement) = {break, if, var, for, while, IDENTIFIER, window.prompt, window, parseFloat, document.writeln, document.write, document, switch, ++, --}

First(*simple-stmt*) = {{, break, if, var, for, while, IDENTIFIER, window.prompt, window, parseFloat, document.writeln, document.write, document, switch, //, ++, --}

First(*if-stmt*) = {if}

First(*declaration-stmt*) = {var}

First(*id*) = {IDENTIFIER}

First(*assign-stmt*) = {IDENTIFIER}

First(*assign-op*) = {=, −=, +=}

First(*exp*) = {(, NUMBER, IDENTIFIER}

First(*logic-op*) = {<, >, <=, >=, ==, !=}

First(*simple-exp*) = {(, NUMBER, IDENTIFIER}

First(*add-op*) = {+, −}

First(*term*) = {(, NUMBER, IDENTIFIER}

First(*mul-op*) = {*, /}

First(*factor*) = {(, NUMBER, IDENTIFIER}

First(*loop-stmt*) = {for, while}

First(*for-parameter*) = { (, NUMBER, IDENTIFIER }

First(*function-stmt*) = {window.prompt, window, parseFloat, document.writeln, document.write, document}

First(*function-keyword*) = {window.prompt, window, parseFloat, document.writeln, document.write, document}

First(*function-parameter*) = { NUMBER, IDENTIFIER, LITERAL }

First(*increment-stmt*) = {++, −−, IDENTIFIER}

First(*increment-op*) = {++, −−}

First(*switch-stmt*) = {switch}

First(*case-part*) = {ε, case}

First(*case-block*) = {case}

First(*case-condition*) = {case}

First(*case-parameter*) = {NUMBER, LITERAL}

First(*default-block*) = {default, ε}

First(*comment*) = {//}

Follow Sets

Follow(Program) = {$}
Follow(stmt-sequence) = {**<script_end>, case, }**}
Follow(comment) = {**nextLine**}
Follow(statement) = {**;**}
Follow(if-stmt) = {**;**}
Follow(*declaration-stmt*) = {**;**}
Follow(*loop-stmt*) = {**;**}
Follow(*assign-stmt*) = {**;**, **,**}
Follow(*function-stmt*) = {**;**}
Follow(*switch-stmt*) = {**;**}
Follow(*increment-stmt*) = {**;**}
Follow(simple-stmt) = {**else, ;**}
Follow(id) = {**;**, **,**}
Follow(assign-op) = {**(, NUMBER, IDENTIFIER**}
Follow(exp) = {**;,)**, **,**}

Follow(simple-exp) = {**<, >, <=, >=, ==, !=,), ;**, **,**, **+**, **-**}

Follow(logic-op) = {**(, NUMBER, IDENTIFIER**}

Follow(term) = {**<, >, <=, >=, ==, !=,), ;**, **,**, *****, / }

Follow(add-op) = {**(, NUMBER, IDENTIFIER**}
Follow(mul-op) = {**(, NUMBER, IDENTIFIER**}

Follow(factor) = {**<, >, <=, >=, ==, !=,), ;**, **,**, *****, / }

Follow(for-parameter) = { **)** }
Follow(function-keyword) = { **(** }
Follow(function-parameter) = { **)** }
Follow(increment-op) = {**IDENTIFIER, ;**}
Follow(case-part) = {**default, }** }
Follow(default-block) = {**}**}
Follow(case-block) = {**case**}
Follow(case-condition) = {**:**}
Follow(case-parameter) = {**:**}