LR(0) Items for LR(1) Parser

Compiler Theory - Programming Project 4 : LR(1) Parser

21400646 Lim Chae Eon

LR(0) Item List

(‘ . ’ ← this is a period symbol)

Program*	→	. Program
Program*	→	Program .
Program	→	. **<script_start>** *stmt-sequence* **<script_end>**
Program	→	**<script_start>** . *stmt-sequence* **<script_end>**
Program	→	**<script_start>** *stmt-sequence* . **<script_end>**
Program	→	**<script_start>** *stmt-sequence* **<script_end>** .
stmt-sequence	→	. *statement* **;** *stmt-sequence*
stmt-sequence	→	*statement* . **;** *stmt-sequence*
stmt-sequence	→	*statement* **;** . *stmt-sequence*
stmt-sequence	→	*statement* **;** *stmt-sequence* .
stmt-sequence	→	. *comment* **nextLine**
stmt-sequence	→	*comment* . **nextLine**
stmt-sequence	→	*comment* **nextLine** .
stmt-sequence	→	.
statement	→	. *if-stmt*
statement	→	*if-stmt* .
statement	→	. *declaration-stmt*
statement	→	*declaration-stmt* .
statement	→	. *loop-stmt*
statement	→	*loop-stmt* .
statement	→	. *assign-stmt*
statement	→	*assign-stmt* .
statement	→	. *function-stmt*
statement	→	*function-stmt* .
statement	→	. *switch-stmt*
statement	→	*switch-stmt* .
statement	→	. *increment-stmt*

statement	→	*increment-stmt* .
statement	→	. **break**
statement	→	**break** .
simple-stmt	→	. *stmt-sequence*
simple-stmt	→	*stmt-sequence* .
simple-stmt	→	. **{** *stmt-sequence* **}**
simple-stmt	→	**{** . *stmt-sequence* **}**
simple-stmt	→	**{** *stmt-sequence* . **}**
simple-stmt	→	**{** *stmt-sequence* **}** .
if-stmt	→	. **if (** *exp* **)** *simple-stmt*
if-stmt	→	**if** . **(** *exp* **)** *simple-stmt*
if-stmt	→	**if (** . *exp* **)** *simple-stmt*
if-stmt	→	**if (** *exp* . **)** *simple-stmt*
if-stmt	→	**if (** *exp* **)** . *simple-stmt*
if-stmt	→	**if (** *exp* **)** *simple-stmt* .
if-stmt	→	. **if (** *exp* **)** *simple-stmt* **else** *simple-stmt*
if-stmt	→	**if** . **(** *exp* **)** *simple-stmt* **else** *simple-stmt*
if-stmt	→	**if (** . *exp* **)** *simple-stmt* **else** *simple-stmt*
if-stmt	→	**if (** *exp* . **)** *simple-stmt* **else** *simple-stmt*
if-stmt	→	**if (** *exp* **)** . *simple-stmt* **else** *simple-stmt*
if-stmt	→	**if (** *exp* **)** *simple-stmt* . **else** *simple-stmt*
if-stmt	→	**if (** *exp* **)** *simple-stmt* **else** . *simple-stmt*
if-stmt	→	**if (** *exp* **)** *simple-stmt* **else** *simple-stmt* .
declaration-stmt	→	. **var** *id*
declaration-stmt	→	**var** . *id*
declaration-stmt	→	**var** *id* .
id	→	. *assign-stmt*
id	→	*assign-stmt* .

id	→	. *id* **,** *id*
id	→	*id* . **,** *id*
id	→	*id* **,** . *id*
id	→	*id* **,** *id* .
id	→	. **IDENTIFIER**
id	→	**IDENTIFIER** .
assign-stmt	→	. **IDENTIFIER** *assign-op exp*
assign-stmt	→	**IDENTIFIER** . *assign-op exp*
assign-stmt	→	**IDENTIFIER** *assign-op* . *exp*
assign-stmt	→	**IDENTIFIER** *assign-op exp* .
assign-op	→	. **=**
assign-op	→	**=** .
assign-op	→	. −=
assign-op	→	−= .
assign-op	→	. **+=**
assign-op	→	**+=** .
exp	→	. *simple-exp logic-op simple-exp*
exp	→	*simple-exp* . *logic-op simple-exp*
exp	→	*simple-exp logic-op* . *simple-exp*
exp	→	*simple-exp logic-op simple-exp* .
exp	→	. *simple-exp*
exp	→	*simple-exp* .
logic-op	→	. **<**
logic-op	→	**<** .
logic-op	→	. **>**
logic-op	→	**>** .
logic-op	→	. **<=**
logic-op	→	**<=** .

logic-op	→	. >=
logic-op	→	>= .
logic-op	→	. ==
logic-op	→	== .
logic-op	→	. !=
logic-op	→	!= .
simple-exp	→	. *simple-exp add-op term*
simple-exp	→	*simple-exp* . *add-op term*
simple-exp	→	*simple-exp add-op* . *term*
simple-exp	→	*simple-exp add-op term* .
simple-exp	→	. *term*
simple-exp	→	*term* .
add-op	→	. +
add-op	→	+ .
add-op	→	. −
add-op	→	− .
term	→	. *term mul-op factor*
term	→	*term* . *mul-op factor*
term	→	*term mul-op* . *factor*
term	→	*term mul-op factor* .
term	→	. *factor*
term	→	*factor* .
mul-op	→	. *
mul-op	→	* .
mul-op	→	. /
mul-op	→	/ .
factor	→	. (*exp*)
factor	→	(. *exp*)

factor	→	**(** *exp* . **)**
factor	→	**(** *exp* **)** .
factor	→	. **NUMBER**
factor	→	**NUMBER** .
factor	→	. **IDENTIFIER**
factor	→	**IDENTIFIER** .
loop-stmt	→	. **for** **(** *for-parameter* **)** *simple-stmt*
loop-stmt	→	**for** . **(** *for-parameter* **)** *simple-stmt*
loop-stmt	→	**for** **(** . *for-parameter* **)** *simple-stmt*
loop-stmt	→	**for** **(** *for-parameter* . **)** *simple-stmt*
loop-stmt	→	**for** **(** *for-parameter* **)** . *simple-stmt*
loop-stmt	→	**for** **(** *for-parameter* **)** *simple-stmt* .
loop-stmt	→	. **while** **(** *exp* **)** *simple-stmt*
loop-stmt	→	**while** . **(** *exp* **)** *simple-stmt*
loop-stmt	→	**while** **(** . *exp* **)** *simple-stmt*
loop-stmt	→	**while** **(** *exp* . **)** *simple-stmt*
loop-stmt	→	**while** **(** *exp* **)** . *simple-stmt*
loop-stmt	→	**while** **(** *exp* **)** *simple-stmt* .
for-parameter	→	. exp **;** exp **;** exp
for-parameter	→	exp . **;** exp **;** exp
for-parameter	→	exp **;** . exp **;** exp
for-parameter	→	exp **;** exp . **;** exp
for-parameter	→	exp **;** exp **;** . exp
for-parameter	→	exp **;** exp **;** exp .
function-stmt	→	. *function-keyword* **(** *function-parameter* **)**
function-stmt	→	*function-keyword* . **(** *function-parameter* **)**
function-stmt	→	*function-keyword* **(** . *function-parameter* **)**
function-stmt	→	*function-keyword* **(** *function-parameter* . **)**

function-stmt	→	*function-keyword* **(** *function-parameter* **)** .
function-keyword	→	. **window.prompt**
function-keyword	→	**window.prompt** .
function-keyword	→	. **window**
function-keyword	→	**window** .
function-keyword	→	. **parseFloat**
function-keyword	→	**parseFloat** .
function-keyword	→	. **document.writeln**
function-keyword	→	**document.writeln** .
function-keyword	→	. **document.write**
function-keyword	→	**document.write** .
function-keyword	→	. **document**
function-keyword	→	**document** .
function-parameter	→	. **IDENTIFIER**
function-parameter	→	**IDENTIFIER** .
function-parameter	→	. **LITERAL**
function-parameter	→	**LITERAL** .
function-parameter	→	. **NUMBER**
function-parameter	→	**NUMBER** .
increment-stmt	→	. *increment-op* **IDENTIFIER**
increment-stmt	→	*increment-op* . **IDENTIFIER**
increment-stmt	→	*increment-op* **IDENTIFIER** .
increment-stmt	→	. **IDENTIFIER** *increment-op*
increment-stmt	→	**IDENTIFIER** . *increment-op*
increment-stmt	→	**IDENTIFIER** *increment-op* .
increment-op	→	. **++**
increment-op	→	**++** .
increment-op	→	. −−

increment-op	→	**−−** .
switch-stmt	→	. **switch (IDENTIFIER) {** *case-part default-block* **}**
switch-stmt	→	**switch** . **(IDENTIFIER) {** *case-part default-block* **}**
switch-stmt	→	**switch (** . **IDENTIFIER) {** *case-part default-block* **}**
switch-stmt	→	**switch (IDENTIFIER** . **) {** *case-part default-block* **}**
switch-stmt	→	**switch (IDENTIFIER)** . **{** *case-part default-block* **}**
switch-stmt	→	**switch (IDENTIFIER) {** . *case-part default-block* **}**
switch-stmt	→	**switch (IDENTIFIER) {** *case-part* . *default-block* **}**
switch-stmt	→	**switch (IDENTIFIER) {** *case-part default-block* . **}**
switch-stmt	→	**switch (IDENTIFIER) {** *case-part default-block* **}** .
case-part	→	. *case-block case-part*
case-part	→	*case-block* . *case-part*
case-part	→	*case-block case-part* .
case-part	→	.
case-block	→	. *case-condition* **:** *stmt-sequence*
case-block	→	*case-condition* . **:** *stmt-sequence*
case-block	→	*case-condition* **:** . *stmt-sequence*
case-block	→	*case-condition* **:** *stmt-sequence* .
case-condition	→	. **case** *case-parameter*
case-condition	→	**case** . *case-parameter*
case-condition	→	**case** *case-parameter* .
case-parameter	→	. **NUMBER**
case-parameter	→	**NUMBER** .
case-parameter	→	. **LITERAL**
case-parameter	→	**LITERAL** .
default-block	→	. **default :** *stmt-sequence*
default-block	→	**default** . **:** *stmt-sequence*
default-block	→	**default :** . *stmt-sequence*

default-block	→	**default :** *stmt-sequence* .
default-block	→	.
comment	→	. **// anything**
comment	→	**//** . **anything**
comment	→	**// anything** .