Especificação da Gramática da Linguagem L−−

Lucas Montenegro e Letícia Medeiros

2020.1

Universidade Federal de Alagoas - Instituto de Computação
Compiladores
Professor Alcino Dall lgna Junior

Conteúdo

1 Analisador Sintático

O analisador sintático utilizado para o desenvolvimento foi o analisador descendente LL(1) preditivo recursivo. Foi realizada apenas uma alteração na GLC e na gramática LL(1) em relação a especificação da linguagem, onde o operador de concatenação é o operador de menor precedência na GLC e na gramática LL(1).

2 Precedência de operadores

A ordem de precedência, que foi alterada para essa GLC e gramática LL(1), vai de cima para baixo e tem a seguinte tabela:

Operador	Ordem de precedência	Associatividade
Negação	‘ ! ’	Direita para esquerda
Operador unário negativo	‘ - ’	Direita para esquerda
Exponenciação	‘ ∧ ’	Direita para esquerda
Operadores aritméticos	‘ ∗ ’ , ‘ / ’ e ‘ % ’	Esquerda para direita
Operadores aritméticos	‘ + ’ e ‘ - ’	Esquerda para direita
Operadores relacionais	‘ < ’ , ‘ <= ’ , ‘ > ’ e ‘ >= ’	Não Associativo
Operadores relacionais	‘==’ , ‘!=’	Não Associativo
Conjunção	‘ && ’	Esquerda para direita
Disjunção	‘ || ‘	Esquerda para direita
Concatenação	’ @ ’	Esquerda para direita

3 Gramática

Segue abaixo as regras de produção da GLC e da gramática LL(1) da linguagem L–, onde S é o não terminal inicial das produções.

3.1 Regras de Produção GLC

S = Decl S
S = Atrib S
S = ’eof’

Decl = DeclVar
Decl = DeclArr
Decl = DeclFunc

DeclVar = Type VarOp ’;’

DeclArr = ’array’ Type ArrOp ’;’

ArrOp = ’id’ ’[’ Expr ’]’ ArrCommaOp

ArrCommaOp = ’,’ ArrOp
ArrCommaOp = ’épsilon’

VarOp = ’id’ CommaOp

CommaOp = ’,’ VarOp
CommaOp = ’épsilon’

DeclFunc = ’function’ TypeOrVoid MainOrId DeclParam Scope

TypeOrVoid = Type | ’void’

MainOrId = ’main’
MainOrId = ’id’

DeclParam = ’(’ ParamOpOrNoParam ’)’

ParamOpOrNoParam = ParamOp | ’épsilon’

ParamOp = DeclVarOp

ParamOp = DeclArrOp

DeclVarOp = Type 'id' ParamCommaOp

DeclArrOp = 'array' Type 'id' '[' ']' ParamCommaOp

ParamCommaOp = ',' ParamOp
ParamCommaOp = 'épsilon'

Scope = '{' Sentences '}'

Sentences = DeclVar Sentences
Sentences = DeclArr Sentences
Sentences = Atrib Sentences
Sentences = FuncCall Sentences
Sentences = Instructions Sentences
Sentences = IfConditional Sentences
Sentences = LogicalLoop Sentences
Sentences = CountLoop Sentences
Sentences = 'break' ';' Sentences
Sentences = 'return' Return ';' Sentences
Sentences = 'épsilon'

FuncCall = 'id' '(' VarOpOrNoVar ')' ';'

VarOpOrNoVar = VarOrArrParam | 'épsilon'

Instructions = 'read' '(' VarOrArrParam ')' ';'
Instructions = 'write' '(' 'constString' WriteParam ')' ';'

WriteParam = ',' VarOrArrParam | 'épsilon'

VarOrArrParam = 'id' VarOrArr VarOrArrCommaOp

VarOrArr = '[' Expr ']' | 'épsilon'

VarOrArrCommaOp = ',' VarOrArrParam
VarOrArrCommaOp = 'épsilon'

IfConditional = 'if' '(' Expr ')' Scope OtherConditional

OtherConditional = ElsifConditional | ElseConditional | 'épsilon'

ElsifConditional = 'elsif' '(' Expr ')' Scope OtherConditional

ElseConditional = 'else' Scope

LogicalLoop = 'while' '(' Expr ')' Scope

CountLoop = 'for' '(' 'int' 'id' ',' 'constInt' ',' 'constInt' ',' 'constInt' ')' Scope

Return = Expr
Return = 'épsilon'

Atrib = 'id' IndiceOp '=' Expr ';'

IndiceOp = '[' Expr ']'
IndiceOp = 'épsilon'

Expr = Expr 'optConcat' Eb
Expr = Eb

Eb = Eb 'optOr' Tb
Eb = Tb

Tb = Tb 'optAnd' Fb
Tb = Fb

Fb = Fb OptRel1 Ra
Fb = 'optNot' Fb
Fb = Ra

Ra = Ra OptRel2 Ea
Ra = Ea

Ea = Ea 'optAdd' Ta
Ea = Ea 'optSub' Ta
Ea = Ta

Ta = Ta 'optMul' Pa
Ta = Ta 'optDiv' Pa
Ta = Ta 'optMod' Pa
Ta = Pa

Pa = Pa 'optPow' Fa
Pa = Fa

Fa = '(' Expr ')'
Fa = 'optSub' Fa
Fa = Variables
Fa = Const

OptRel1 = '<=' | '>=' | '<' | '>'

OptRel2 = '==' | '!='

Variables = 'id' | 'id' '[' Expr ']' | 'id' '(' VarOpOrNoVar ')'

Const = 'constInt' | 'constFloat' | 'constBool' | 'constChar' | 'constString'

Type = 'int' | 'float' | 'bool' | 'char' | 'string'

3.2 Regras de Produção LL(1)

S = Decl S
S = Atrib S
S = 'eof'

Decl = DeclVar
Decl = DeclArr
Decl = DeclFunc

DeclVar = Type VarOp ';'

DeclArr = 'array' Type ArrOp ';'

ArrOp = 'id' '[' Expr ']' ArrCommaOp

ArrCommaOp = ',' ArrOp
ArrCommaOp = 'épsilon'

VarOp = 'id' CommaOp

CommaOp = ',' VarOp
CommaOp = 'épsilon'

DeclFunc = 'function' TypeOrVoid MainOrId DeclParam Scope

TypeOrVoid = Type | 'void'

MainOrId = 'main'
MainOrId = 'id'

DeclParam = '(' ParamOpOrNoParam ')'

ParamOpOrNoParam = ParamOp | 'épsilon'

ParamOp = DeclVarOp
ParamOp = DeclArrOp

DeclVarOp = Type 'id' ParamCommaOp

DeclArrOp = 'array' Type 'id' '[' ']' ParamCommaOp

ParamCommaOp = ',' ParamOp
ParamCommaOp = 'épsilon'

Scope = '{' Sentences '}'

Sentences = DeclVar Sentences

Sentences = DeclArr Sentences
Sentences = AtribOrFuncCall Sentences
Sentences = Instructions Sentences
Sentences = IfConditional Sentences
Sentences = LogicalLoop Sentences
Sentences = CountLoop Sentences
Sentences = 'break' ';' Sentences
Sentences = 'return' Return ';' Sentences
Sentences = 'épsilon'

AtribOrFuncCall = 'id' CheckAtribOrFuncCall

CheckAtribOrFuncCall = '[' AtribOp1
CheckAtribOrFuncCall = '=' AtribOp2
CheckAtribOrFuncCall = '(' FuncCall

AtribOp1 = Expr ']' '=' Expr ';'

AtribOp2 = Expr ';'

FuncCall = VarOpOrNoVar ')' ';'

VarOpOrNoVar = VarOrArrParam | 'épsilon'

Instructions = 'read' '(' VarOrArrParam ')' ';'
Instructions = 'write' '(' 'constString' WriteParam ')' ';'

WriteParam = ',' VarOrArrParam | 'épsilon'

VarOrArrParam = 'id' VarOrArr VarOrArrCommaOp

VarOrArr = '[' Expr ']' | 'épsilon'

VarOrArrCommaOp = ',' VarOrArrParam
VarOrArrCommaOp = 'épsilon'

IfConditional = 'if' '(' Expr ')' Scope OtherConditional

OtherConditional = ElsifConditional | ElseConditional | 'épsilon'

ElsifConditional = 'elsif' '(' Expr ')' Scope OtherConditional

ElseConditional = 'else' Scope

LogicalLoop = 'while' '(' Expr ')' Scope

CountLoop = 'for' '(' 'int' 'id' ',' 'constInt' ',' 'constInt' ',' 'constInt' ')' Scope

Return = Expr
Return = 'épsilon'

Atrib = 'id' IndiceOp '=' Expr ';'

IndiceOp = '[' Expr ']'
IndiceOp = 'épsilon'

Expr = Eb Exprl

Exprl = 'optConcat' Eb Exprl
Exprl = 'épsilon'

Eb = Tb Ebl

Ebl = 'optOr' Tb Ebl
Ebl = 'épsilon'

Tb = Fb Tbl

Tbl = 'optAnd' Fb Tbl
Tbl = 'épsilon'

Fb = 'optNot' Fb Fbl
Fb = Ra Fbl

Fbl = OptRel1 Ra Fbl
Fbl = 'épsilon'

Ra = Ea Ral

Ral = OptRel2 Ea Ral
Ral = 'épsilon'

Ea = Ta Eal

Eal = 'optAdd' Ta Eal
Eal = 'optSub' Ta Eal
Eal = 'épsilon'

Ta = Pa Tal

Tal = 'optMul' Pa Tal
Tal = 'optDiv' Pa Tal
Tal = 'optMod' Pa Tal
Tal = 'épsilon'

Pa = Fa Pal

Pal = 'optPow' Fa Pal
Pal = 'épsilon'

Fa = '(' Expr ')'
Fa = 'optSub' Fa
Fa = Variables
Fa = Const

OptRel1 = '<=' | '>=' | '<' | '>'

OptRel2 = '==' | '!='

Variables = 'id' VarOrArrOrFunc

VarorArrOrFunc = '[' Expr ']' | '(' VarOpOrNoVar ')' | 'épsilon'

Const = 'constInt' | 'constFloat' | 'constBool' | 'constChar' | 'constString'

Type = 'int' | 'float' | 'bool' | 'char' | 'string'