UNIVERSIDADE FEDERAL DE ALAGOAS

Trabalho de compiladores: Toco

João Lucas Marques Correia

Myron David Lucena Campos Peixoto

Abril - 2019

Sumário

1 Modificações nas categorias dos tokens

Com o propósito de tornar a gramática da linguagem mais simples e transparente, algumas classes de tokens foram removidos conforme **Tabela 2** e outros foram agrupados conforme **Tabela 1**.

Categoria simbólica	**Categorias simbólicas eliminadas**	**Expressão regular**
OPE_LOG	OPE_OU, OPE_E	'||' | '&&'
OPE_REL	OPE_IGU, OPE_DIF, OPE_MEN, OPE_MAI, OPE_MEN_IGU, OPE_MAI_IGU	'==' | '!=' | '<=' | '>= | '<' | '>'
OPE_ADI	OPE_ADI, OPE_SUB_NEG	'+' | '-'
OPE_MULT	OPE_MUL, OPE_DIV	'*'|'/'
OPE_LIM	OPE_FOR_CAM, OPE_FOR_DEC	'%' | '%%'

Tabela 1: Modificações nas categorias de tokens

Categoria simbólica eliminada
MAIN

Tabela 2: Categorias removidas

A categoria main foi removida, agora todos os nomes de função são classificados apenas como tokens do tipo *IDENTIFICADOR*.

2 Gramática

2.1 Gramática original

Listing 1: Exemplo estrutural de um programa Toco

```
Program = LDecl

LDecl = Ldecl Decl
LDecl = Decl

Decl = DeclVar
Decl = DeclFun
```

```
DeclVar = 'var' Type 'id' Atr ';'
DeclVar = 'var' Type 'id' '[' ExpArit ']' Atr ';'

Atr = '=' ExpConcat
Atr = epsilon

DeclFun = 'fun' TypeF 'id' '(' LParam ')''{' LSent '}'

TypeF = 'void'
TypeF = Type

Type = 'boolean'
Type = 'float'
Type = 'int'
Type = 'char'
Type = 'string'

LParam = LParam ',' Param
LParam = Param
LParam = epsilon

Param = Type 'id' '[' ']'
Param = Type 'id'

LSent = LSent Sent
Sent = DeclVar
Sent = Command

Command = 'continue' ';'
Command = 'break' ';'
Command = 'id' '[' ExpArit ']' Atr ';'
Command = 'id' '(' LArg ')' ';'
Command = 'id' Atr ';'
Command = Print ';'
Command = Read ';'
Command = For
Command = While
Command = If
Command = Return
```

```
LArg = LArg ',' Arg
LArg = epsilon
LArg = Arg

Arg = ExpConcat

Print = 'print' '(' ExpConcat ')'

Read = 'read' '(' LParamR ')'

LParamRead = LParamRead ',' ParamRead
LParamRead = ParamRead

ParamRead = 'id'

For = 'for' 'id' 'in' ExpArit 'to' ExpArit Step '{'LSent'}'

Step = 'step' ExpArit
Step = epsilon

While = 'while' '(' ExpBool ')' '{' LSent '}'

If = 'if' '(' ExpBool ')' '{'LSent'}' Else

Else = 'else' '{' LSent '}'
Else = epsilon

Return = 'return' ExpConcat ';'

ExpConcat = ExpConcat '++' ExpBool
ExpConcat = ExpBool

ExpBool = ExpBool 'ope_log' TermBool
ExpBool = TermBool

TermBool = '!' TermBool
TermBool = TermBool 'ope_rel' ExpArit
TermBool = ExpArit
```

```
ExpArit = ExpArit 'ope_adi' TermArit
ExpArit = TermArit

TermArit = TermArit 'ope_mult' ExpLim
TermArit = ExpLim

ExpLim = ExpLim 'ope_lim' FatArit
ExpLim = FatArit

FatArit = 'id' '(' LArg ')'
FatArit = 'id' '[' ExpArit ']'
FatArit = '-' FatArit
FatArit = '(' ExpBool ')'
FatArit = 'id'
FatArit = 'cte_int'
FatArit = 'cte_float'
FatArit = 'cte_char'
FatArit = 'cte_cad_ch'
FatArit = 'cte_bool'
```

2.2 Gramática LL(1)

```
Program =  LDecl

LDecl = Decl LDeclR
LDeclR = Decl LDeclR
LDeclR = epsilon

Decl =  DeclVar
Decl =  DeclFun

DeclVar = 'var' Type 'id' DeclVarAux ';'
DeclVarAux = '[' ExpArit ']' Atr
DeclVarAux = Atr

Atr = '=' ExpConcat
Atr = epsilon
```

```
DeclFun = 'fun' TypeF 'id' '(' LParam ')''{' LSent '}'

TypeF = 'void'
TypeF = Type
Type =  'boolean'
Type = 'float'
Type = 'int'
Type = 'char'
Type = 'string'

LParam = Param LParamR
LParam = epsilon

LParamR = ',' Param LParamR
LParamR = epsilon

Param = Type 'id' ParamR

ParamR = '[' ']'
ParamR = epsilon

LSent = Sent LSent
LSent = epsilon

Sent =  DeclVar
Sent = Command

Command = 'continue' ';'
Command = 'break' ';'
Command = 'id' CommandR ';'
Command = Print ';'
Command = Read ';'
Command = For
Command = While
Command = If
Command = Return

CommandR ='[' ExpArit ']' Atr
CommandR ='(' LArg ')'
CommandR = Atr
```

```
LArg =  Arg LArgR
LArg = epsilon

LArgR = ',' Arg LArgR
LArgR = epsilon

Arg = ExpConcat

Print = 'print' '(' ExpConcat ')'
Read =  'read' '(' LParamRead ')'

LParamRead = ParamRead LParamReadR
LParamReadR = ',' ParamRead LParamReadR
LParamReadR = epsilon
ParamRead = 'id' ParamReadR
ParamReadR ='[' ExpArit ']'
ParamReadR = epsilon

For = 'for' '(' 'id' 'in' ExpArit 'to' ExpArit Step ')' '{'LSent'}'

Step =  'step' ExpArit
Step = epsilon

While = 'while' '(' ExpBool ')' '{' LSent '}'

If = 'if' '(' ExpBool ')' '{'LSent'}' Else

Else =  'else' '{' LSent '}'
Else = epsilon

Return = 'return' ExpConcat ';'

ExpConcat = ExpBool ExpConcatR
ExpConcatR = '++' ExpBool ExpConcatR
ExpConcatR = epsilon

ExpBool = TermBool ExpBool1
ExpBoolR = 'opr_log' TermBool ExpBoolR
ExpBoolR = epsilon
```

```
TermBool = '!' TermBool
TermBool = ExpArit TermBoolR
TermBoolR = 'opr_rel' ExpArit TermBoolR
TermBoolR = epsilon

ExpArit =  TermArit ExpAritR
ExpAritR = 'opa_adi' TermArit ExpAritR
ExpAritR = epsilon

TermArit = ExpLim TermAritR
TermAritR = 'opa_mult' ExpLim TermAritR
TermAritR = epsilon

ExpLim = FatArit ExpLimR
ExpLimR = 'opa_lim' FatArit ExpLimR
ExpLimR = epsilon

FatArit = 'id' FatAritR
FatArit = 'opa_nega' FatArit
FatArit = 'cte_int'
FatArit = 'cte_float'
FatArit = 'cte_char'
FatArit = 'cte_cad_ch'
FatArit = 'cte_bool'

FatAritR = '(' LArg ')'
FatAritR = '[' ExpArit ']'
FatAritR = epsilon
```