PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Compiladores

Alyson Deives Pereira

COMPILADOR PARA LINGUAGEM GL

Descrição do alfabeto, padrão de formação dos lexemas, AFD, Gramática LL(1) e Esquema de Tradução

Professor Alexei Machado

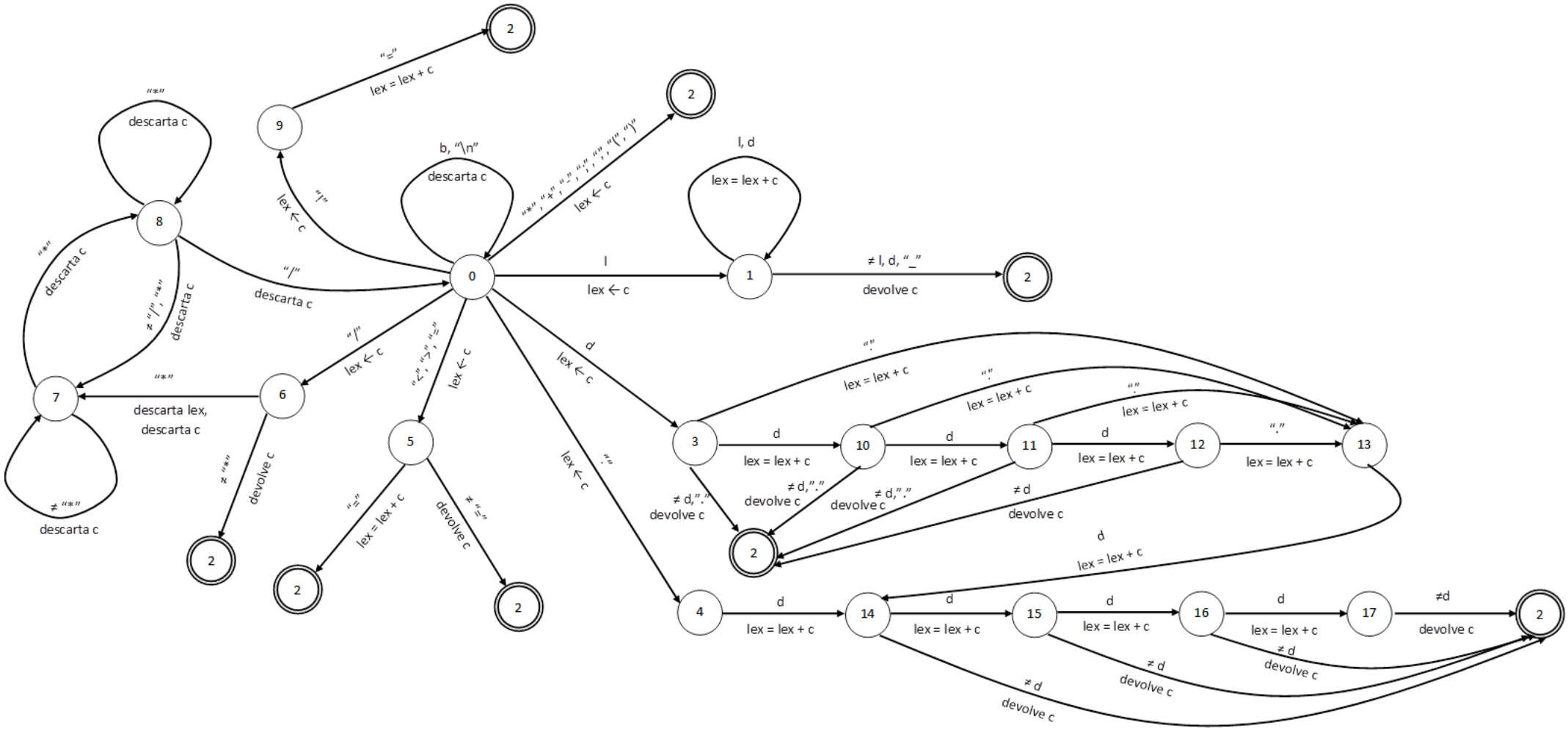
Belo Horizonte

2013

Descrição do Alfabeto e Padrão de Formação dos Lexemas

∑	Lexema
ID	l (l ∪ d)
CONSTANT	(d ∪ d^2 ∪ d^3 ∪ d^4) ∪ (d ∪ d^2 ∪ d^3 ∪ d^4) “.” ∪ (“.” (d ∪ d^2 ∪ d^3 ∪ d^4)) ∪ ((d ∪ d^2 ∪ d^3 ∪ d^4) “.” (d ∪ d^2 ∪ d^3 ∪ d^4))
COMMA	“,”
SEMICOLON	“;”
LEFT_PARENS	“(”
RIGHT_PARENS	“)”
ASSIGN	“=”
SUM	“+”
SUB	“-”
MULT	“*”
DIV	“/”
AND	“and”
OR	“or”
NOT	“not”
EQ	“==”
NEQ	“!=”
LT	“<”
GT	“>”
LE	“<=”
GE	“<=”
BEGIN	“begin”
COLOR	“color”
CONST	“const”
DECLARE	“declare”
ELSE	“else”
END	“end”
FACE	“face”
IF	“if”
INTEGER	“integer”
LIGHT	“light”
OBJECT	“object”
PAUSE	“pause”
POINT	“point”
REAL	“real”
ROTTRANS	“rottrans”
SCALE	“scale”
THEN	“then”
VAR	“var”
WHILE	“while”

Desenho do AFD



Gramática LL(1)

START → declare DECLARATION* COMMAND_BLOCK ENDFILE
DECLARATION → point (1) { id(10)(60)(61) = "(" [+ | -(62)] num(63), [+ | -(62)] num(63) , [+ | -(62)] num(63) ")"; }+ |
light (2){ id(10)(60)(61) = "(" [+ | -(62)] num(63) , [+ | -(62)] num(63) , [+ | -(62)] num(63) ")"; }+ |
color (3){ id(10)(60) = num(18)(64), num1(19)(64), num2(20)(64), num3(21)(64), num4(22)(64) , num5(23)(64) ,
num6(24)(64) ; }+ |
face (4){ id(10)(60) = id1(53) (11)(65) , id2(53)(12)(66) , id3(53)(13)(66) , id4(53)(14)(66) { , id5(53)(15)(66) }* (67); }+ |
object (5){ id(10)(60) = id1(53)(16)(68) { , id2(53)(17)(68) }* (69); }+ |
var (6){ (integer(8) | real(9)) id(10)(25)(60) { , id1(10)(26)(60) }* ; }+ |
const (7){ id(10)(60)(61) = [+ | -(62)] num(27)(60)(70) ; }+
COMMAND_BLOCK → begin COMMAND* end
COMMAND → id (53)(28)(72) = EXP (29); |
while EXP (30)(73)(COMMAND_BLOCK | COMMAND) |
if EXP (30)(74) then (COMMAND_BLOCK | COMMAND)
[else (COMMAND_BLOCK | COMMAND)] |
scale id (53)(31)(75), EXP (32); |
pause EXP (32)(76) ; |
light id (53)(33)(77); |
rottrans id (53)(31) , EXP1(34) , EXP2(35) , EXP3(36) , EXP4(37) , EXP5(38) , EXP6(39)(78) ;
LOGIC_COMPARATION → == (40) | != (40)| < (40) | > (40) | <= (40) | < (40)
EXP → EXP_SUM (41) [LOGIC_COMPARATION (42) EXP_SUM1(43)]
EXP_SUM → [+ | -] EXP_PRODUCT (44) { (+(54) | - (55)| or(56)) EXP_PRODUCT1 (45)}*
EXP_PRODUCT → EXP_VALUE (46){ (*(57) | / (58) | and (59)) EXP_VALUE1 (47) }*
EXP_VALUE → not (48) EXP_VALUE1 (49) | “(” EXP “)” (50) | id (53)(51)(71) | const (52)

Ações semânticas para verificação de tipos e unicidade

1. DECLARATION.*classe* := *classe-ponto*
2. DECLARATION.*classe* := *classe-luz*
3. DECLARATION.*classe* := *classe-cor*
4. DECLARATION.*classe* := *classe-face*
5. DECLARATION.*classe := classe-objeto*
6. DECLARATION.*classe* := *classe-var*
7. DECLARATION.*classe* := *classe-const*
8. DECLARATION.*tipo* := *tipo-inteiro*
9. DECLARATION.*tipo* := *tipo-real*
10. { se id.posiçãoclasse = *vazio* então id.posiçãoclasse := *DECLARATION.classe* senão ERRO }
11. { se id1.posiçãoclasse != *classe-cor* então ERRO }
12. { se id2.posiçãoclasse != *classe-ponto* então ERRO }
13. { se id3.posiçãoclasse != *classe-ponto* então ERRO }
14. { se id4.posiçãoclasse != *classe-ponto* então ERRO }
15. { se id5.posiçãoclasse != *classe-ponto* então ERRO }
16. { se id1.posiçãoclasse != *classe-face* então ERRO }
17. { se id2.posiçãoclasse != *classe-face* então ERRO }
18. { se num.tipo != *tipo-inteiro* então ERRO }
19. { se num1.tipo != *tipo-inteiro* ou num1.value > 63 então ERRO }
20. { se num2.tipo != *tipo-inteiro* ou num2.value > 63 então ERRO }
21. { se num3.tipo != *tipo-inteiro* ou num3.value > 63 então ERRO }
22. { se num4.tipo != *tipo-inteiro* ou num4.value > 63 então ERRO }
23. { se num5.tipo != *tipo-inteiro* ou num5.value > 63 então ERRO }
24. { se num6.tipo != *tipo-inteiro* ou num6.value > 63 então ERRO }
25. { se id.posiçãotipo = *tipo-vazio* então id.posiçãotipo := *DECLARATION.tipo* senão ERRO }
26. { se id1.posiçãotipo = *tipo-vazio* então id.posiçãotipo := *DECLARATION.tipo* senão ERRO }
27. { se id.posiçãotipo = *tipo-vazio* então id.posiçãotipo = *num.tipo* senão ERRO}
28. { se id.posiçãoclasse != *classe-var* então ERRO }
29. { se *EXP.tipo* = *tipo-lógico* então ERRO senao se id.posiçãotipo = *tipo-inteiro* && *EXP.tipo* = *tipo-real* então ERRO }
30. { se *EXP.tipo* != *tipo-lógico* então ERRO }
31. { se id.posiçãoclasse != *classe-objeto* então ERRO }

32. { se *EXP.tipo = tipo-lógico* então ERRO }
33. { se id.posiçãoclasse != *classe-luz* então ERRO }
34. { se EXP_1*.tipo = tipo-lógico* então ERRO }
35. { se EXP_2*.tipo = tipo-lógico* então ERRO }
36. { se EXP_3*.tipo = tipo-lógico* então ERRO }
37. { se EXP_4*.tipo = tipo-lógico* então ERRO }
38. { se EXP_5*.tipo = tipo-lógico* então ERRO }
39. { se EXP_6*.tipo = tipo-lógico* então ERRO }
40. { *LOGIC_COMPARATION.tipo = tipo-lógico* }
41. { *EXP.tipo = EXP_SUM.tipo* }
42. { *EXP.tipo = tipo-lógico* }
43. { se *EXP_SUM.tipo = tipo-lógico* então ERRO}
44. { *EXP_SUM.tipo = EXP_PRODUCT.tipo* }
45. { se EXP_SUM.op == OR and !(*EXP_PRODUCT.tipo == tipo-lógico* && *EXP_PRODUCT*$_1$*.tipo == tipo-lógico)* então ERRO senão *EXP_SUM.tipo = EXP_PRODUCT*$_1$*.tipo* }
46. { *EXP_PRODUCT.tipo = EXP_VALUE.tipo* }
47. { se *EXP_PRODUCT.op* == AND && !(*EXP_VALUE*$_1$*.tipo == tipo-lógico* && *EXP_VALUE*$_1$*.tipo == tipo-lógico)* então ERRO senão se *EXP_PRODUCT.op* = DIV então *EXP_PRODUCT.tipo = tipo-real* senão *EXP_PRODUCT.tipo = EXP_VALUE*$_1$*.tipo*}
48. { *EXP_VALUE.tipo = tipo-lógico* }
49. { se *EXP_VALUE*$_1$*.tipo* != *tipo-lógico* então ERRO}
50. { *EXP_VALUE.tipo = EXP.tipo*}
51. { *EXP_VALUE.tipo* = id.posiçãotipo}
52. { *EXP_VALUE.tipo* = num.tipo}
53. { se id.posiçãoclasse == *classe-vazia* então ERRO }
54. {*EXP_SUM.op* = SUM}
55. {*EXP_SUM.op* = SUB}
56. {*EXP_SUM.op* = OR}o
57. {*EXP_PRODUCT.op* = MULT}
58. {*EXP_PRODUCT.op* = DIV}
59. {*EXP_PRODUCT.op* = AND}

Ações semânticas para geração de código

60. {id.posiçãoend = DS. DS+= tamanho(id)}
61. {*DECLARATION.sinal* = 1}
62. {*DECLARATION.sinal* = -1}
63. {STIF *#DECLARATION.sinal * num.val* DS}
64. {STI *#num.val* DS}
65. {*DECLARATION.endCor* = id$_1$.posiçãoend}
66. {*DECLARATION.listaEndPonto*.inserir(id$_x$.posiçãoend)}
67. {STI *DECLARATION.listaEndFacePonto*.tamanho DS
    STI *DECLARATION.endCor* DS
    Para cada *endereco* em *listaEndPonto* faça
    STI *endereco* DS}
68. {*DECLARATION.listaEndFace*.inserir(id$_x$.posiçãoend)}
69. {STI *DECLARATION.listaEndFacePonto*.tamanho DS
    STI *#1.0* DS
    Para cada *endereco* em *listaEndFace* faça
    STI *endereco* DS}
70. {se num.tipo == inteiro então STI sinal*num.val DS
    senão STIF sinal*num.val DS}
71. {EXP_VALUE.end = id.posicaoend}

72.
73. {
se *EXP*.tipo == TIPO_INTEIRO então
LOD A, *EXP*.endereco(DS)
senão
LODF AX, *EXP*.endereco(DS)

se id.posicaotipo == TIPO_REAL então
se *EXP*.tipo == TIPO_INTEIRO então
CNV AX, A
STOF AX, id.posicaoend(DS)
senão
STO A, id.posicaoend(DS)
}
73. {
rotInicio = novoRot()
rotFim = novoRot()
rotInicio:
se *EXP*.tipo == TIPO_LOGICO então
LOD A, *EXP*.endereco(DS)
BZR A,RotFim
(COMMAND_BLOCK | COMMAND)
JMP rotInicio
rotFim:
}
74. {
rotFalso = novoRot()
rotFim = novoRot()
LOD A, *EXP*.endereco(DS)
BZR A, rotFalso
(COMMAND_BLOCK | COMMAND)
se currentToken.token == ELSE então
JMP rotFim
rotFalso:
(COMMAND_BLOCK | COMMAND)
rotFim:
senão
rotFim:
}
75. {
se *EXP*.tipo == TIPO_REAL então
LODF AX, *EXP*.endereco(DS)
senão
LOD B, *EXP*.endereco(DS)
CNV AX, B
LDI A, #id.posicaoend
ESC A, AX
}
76. {
se *EXP*.tipo == TIPO_REAL então
LODF AX, *EXP*.endereco(DS)
senão
LOD B, EXP.endereco(DS)
CNV AX, B
TME AX
}

77. {
    LOD A, id.posicaoend
    LGT A
}
78. {
    se EXP_1.tipo == TIPO_REAL então
        LODF AX, EXP_1.endereco(DS)
    senão
        LOD B, EXP_1.endereco(DS)
        CNV AX, B

    se EXP_2.tipo == TIPO_REAL então
        LODF BX, EXP_2.endereco(DS)
    senão
        LOD B, EXP_2.endereco(DS)
        CNV BX, B

    se EXP_3.tipo == TIPO_REAL então
        LODF CX, EXP_3.endereco(DS)
    senão
        LOD B, EXP_3.endereco(DS)
        CNV CX, B

    se EXP_5.tipo == TIPO_REAL então
        LODF DX, EXP_5.endereco(DS)
    senão
        LOD B, EXP_5.endereco(DS)
        CNV DX, B

    se EXP_5.tipo == TIPO_REAL então
        LODF EX, EXP_5.endereco(DS)
    senão
        LOD B, EXP_5.endereco(DS)
        CNV EX, B

    se EXP_6.tipo == TIPO_REAL então
        LODF FX, EXP_6.endereco(DS)
    senão
        LOD B, EXP_6.endereco(DS)
        CNV FX, B

    LDI A, id.posicaoend
    RTR
}