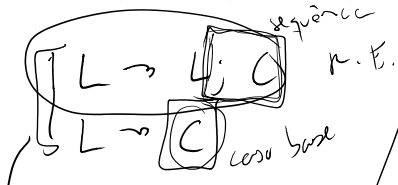


LISTA -> LISTA ; CMD LISTACMD
 CMD -> id := EXP
 EXP -> id () num

De <<http://www.dcc.ufri.br/~fabiom/comp/lista1.html>>



[C -> id := EXP
 p. C.



EBNF
 $E \rightarrow id (E | '()')^L \rightarrow C$
 $\rightarrow C(L' \rightarrow C)$

E, N, E
 $L \rightarrow CL'$
 $L' \rightarrow ; CL'$
 $L' \rightarrow$

CFL
 $E \rightarrow id E'$
 $E' \rightarrow ()$
 $E' \rightarrow m$
 FATORAÇÃO À ESQUERDA

$L \rightarrow C; L$
 $L \rightarrow C$
 $L \rightarrow CL'$
 $L' \rightarrow$
 $L' \rightarrow ; L$

$E \rightarrow E + E'$
 $E \rightarrow E - E'$
 $E \rightarrow E'$
 $E' \rightarrow m$
 $E' \rightarrow id$
 $E' \rightarrow (E)$

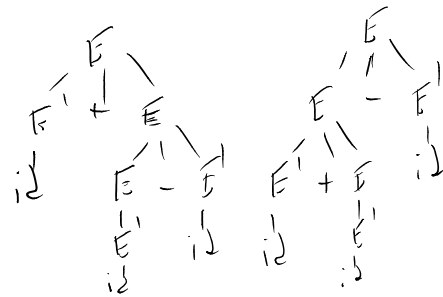
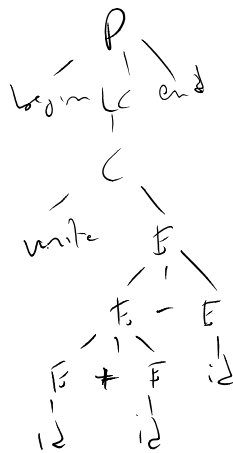
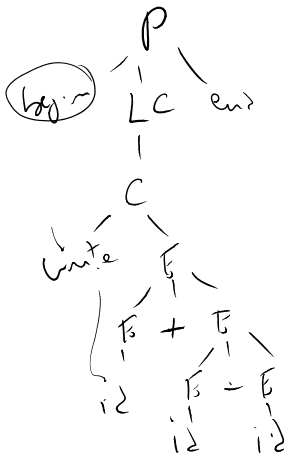
id L id - id
AMBIGUO

$E \rightarrow (E) + E$
 $E \rightarrow E - (E)$
 $E \rightarrow E'$
 $E' \rightarrow m$
 $E' \rightarrow id$
 $E' \rightarrow (E)$

PROGRAMA -> begin LISTACMD end
 LISTACMD -> LISTACMD CMD
 | CMD
 CMD -> do id := num to num begin LISTACMD end
 | read id
 | write EXP
 | id := EXP
 EXP -> EXP + EXP | EXP - EXP | num | id | (EXP)

De <<http://www.dcc.ufri.br/~fabiom/comp/lista1.html>>
 begin write id + id - id end

begin write id + id - id end



9.

A -> AA | (A) | *vazio*

De <<http://www.dcc.ufri.br/~fabiom/comp/lista1.html>>

vazio

vazio





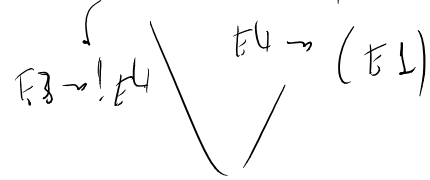
A

11. Escreva uma gramática para expressões booleanas contendo as constantes true e false e os operadores &&, || e !, além de parênteses. O operador || tem precedência menor que &&, e este menor que !. A gramática não pode ser ambígua.

Ⓜ < && < !

De <<http://www.dcc.ufrj.br/~fabiom/comp/lista1.html>>

* $E \rightarrow E || E$ $E \rightarrow E \&\& E$ $E \rightarrow !E$ $E \rightarrow true$
 $E \rightarrow false$



e $E \rightarrow !E$

$E \rightarrow !E$
 $E \rightarrow true$
 $E \rightarrow false$
 $E \rightarrow (E)$

! true || false

- 5 + 3

14.

```
PROGRAMA -> begin LISTACMD end
LISTACMD -> LISTACMD CMD
          | CMD
CMD -> do id := num to num begin LISTACMD end
      | read id
      | write EXP
      | id := EXP
EXP -> EXP + EXP | EXP * EXP | num | id | ( EXP )
```

De <<http://www.dcc.ufrj.br/~fabiom/comp/lista1.html>>

$LC \rightarrow C LC'$ *simplificação opcional*
 $LC' \rightarrow C LC'$
 $LC' \rightarrow$

$E \rightarrow num E'$
 $E \rightarrow id E'$
 $E \rightarrow (E) E'$

$E' \rightarrow + E E'$
 $E' \rightarrow - E E'$
 $E' \rightarrow$

$First+(A \rightarrow) = Follow(A) = \{ \$ \}$

$E \rightarrow E + T$
 $E \rightarrow E - T$
 $E \rightarrow T$
 $T \rightarrow n$
 $T \rightarrow id$
 $T \rightarrow (E)$
 $E \rightarrow T E'$
 $E' \rightarrow + T E'$
 $E' \rightarrow - T E'$
 $E' \rightarrow$

16. Dada a gramática $A \rightarrow (A)A | \text{*vazio*}$, escreva pseudocódigo para analisá-la de forma recursiva com retrocesso local e sem retrocesso (LL(1)).

De <<http://www.dcc.ufrj.br/~fabiom/comp/lista1.html>>

Arvore A() { // retrocesso local

```

Anvove res = new Anvove();
int atual = pos;
try {
    Anvove n = new Anvove();
    n.filho (match ('c'));
    n.filho (A());
    n.filho (match ('|'));
    n.filho (A());
    res.addAll(n.filhos);
} catch (Falha f) {
    pos = atual;
}
return res;
}

```

```

Anvove A() { // sem necessidade
    Anvove res = new Anvove();
    if (ca == 'c') {
        res.filho (match ('c'));
        res.filho (A());
        res.filho (match ('|'));
        res.filho (A());
    } // else {}
    return res;
}

```