**Aula 02/06/2020**

Gramática Livre de Contexto G=(N, T, P, S)

N = {E}

T = {num,+}

P = { E -> E + E , E -> num }

S = E

num = qualquer digito: 0, 1, 2, 3, 4, ... 9

Derivando a palavra 1+2+3 com derivações sucessivas.

E=>E+E=>E+E+E=>E+E+num=>E+E+3=>E+num+3=>E+2+3=>num+2+3=>1+2+3

Derivando a palavra 1+2+3 com árvore de derivação ou árvore sintática

folhas da arvore com a palavra gerada

3

2

num

num

1

num

E + E

vértices internos da árvore

E + E

raiz da árvore com o símbolo inicial da gramática

E

**Aula 03/06/2020**

Considere a linguagem L = { W ∈ {a,b}\*| W=anbn e n>=0}

N = { S }

T = { a, b }

P = {S → aSb, S → ε }

S = S

aabb

Considere a linguagem L = { w ∈ {a,b}\*| w=anbm e m ≠ n e n>=0 e m >=0 }

N = { S }

T = { a, b }

P = {S → aSb, S → ε }

S = T

(N U T)\*

T → AS, T → SB

A→ aA , A→ a

B→ bB, B→ b

S → ab, S → ε

W= aaab

T=> AS => AaSb => Aab => aAab => aaab

W=a

T=>AS => Avazio => a

Ambiguidade

N = {E}

T = {a,b,c,+}

P = {E→E+E, E→a, E→b, E→c }

S = E

W=a+b+c

Ambiguidade

N = {E}

T = {1,2,3,+,\*}

P = {E→E+E, E→E\*E, E→1, E→2, E→3 }

S = E

W=1+2\*3

resultado=9 resultado=7

A segunda árvore é a que deve ser gerada em uma árvore de expressão, pois o resultado correto da expressão 1+2\*3=7

**Aula 09/06/2020**

Ambiguidade

N = {E}

T = {1,2,3,+,\*}

P = {E→E+E, E→E\*E, E→1, E→2, E→3 }

S = E

W = 1

E=> 1

**Eliminando a ambiguidade (Sugestão Bruno Godoi)**

E->E+N, E->E\*N, E->N, N->1, N->2, N->3

W=1

E=>N=>1

Para a palavra 1+2\*3

a gramática acima gera uma árvore que

quando for implementada por um compilador

irá gerar uma conta errada, pois os operadores

mais próximo das folhas são resolvidos primeiro,

assim a expressão fará a soma (1+2)=3 e depois

a multiplicação 3\*3=9

**Eliminando a ambiguidade (solução Fabio Lubacheski)**

Para a gramática com as regras E→E+E, E→E\*E, E→1, E→2, E→3, o ideal é que o operador de multiplicação tenha prioridade em relação ao operador de adição.

Além disso quando tivermos em uma expressão com operadores iguais, ou seja, um operando entre dois operadores iguais: 1+**2**+3, o operando **2** está entre dois operadores de adição, o operando se associa ao operador mais à esquerda, ou seja, resolvemos a expressão da esquerda para direita.

Para eliminar a ambiguidade devemos retirar do mesmo não terminal os operadores de adição e multiplicação, sendo E o não terminal inicial da gramática

E→E+T, E→T

T→ T\*F, T→F

F→1, F→2, F→3

Palavras aceitas

1

1+2+...+3

1\*2\*...\*3

1+2\*3

1\*2+3

A gramática abaixo pode ser reescrita para notação BNF (***Backus-Naur Form***)

E→E+T, E→T

T→ T\*F, T→F

F→1, F→2, F→3

Notação BNF

E ::= E+T | T

T::= T\*F | F

F::= 1 | 2 | 3



**Ambiguidade do else pendente**

CMD ::= *a* | *b* | if ( *e* ) then CMD else CMD |

if ( *e* ) then CMD

**1a palavra**

if ( e ) then

if ( e ) then

a

else

b

**2a palavra**

if ( e ) then

if ( e ) then

a

else

b

Qual das suas árvores é a derivação correta para as palavras acima ?considere que as palavras acima são iguais.

Nas linguagens de programação o else sempre se associa ao if mais próximo, ou seja, a primeira árvore sempre deve ser gerada.