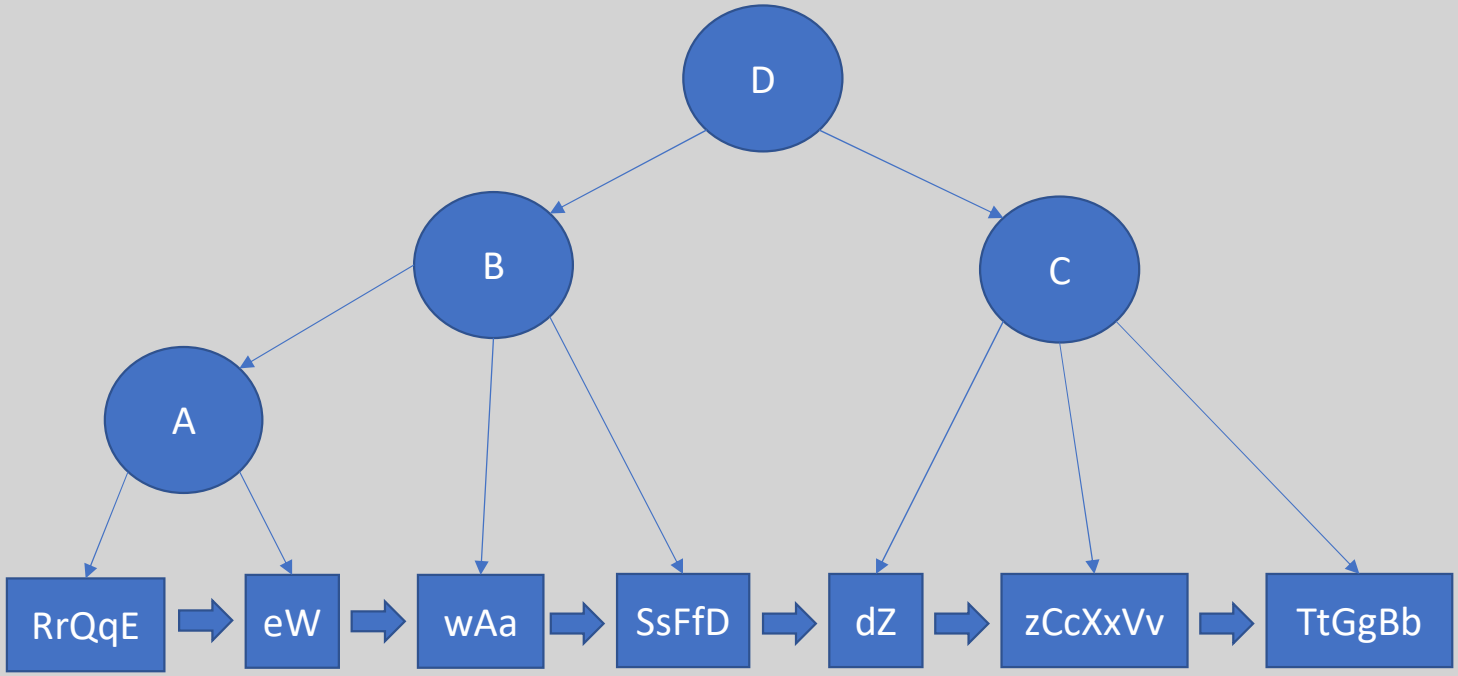
Análise Sintática - Objetivos

Agrupa os tokens fornecidos pelo analisador léxico em estruturas sintáticas, construindo uma árvore sintática.



Análise Sintática – Regras Sintáticas

Utiliza uma série de regras de sintaxe, presentes na gramática Livre de Contexto da linguagem fonte.

[D] ➡ [B] [C] | [C]

[C] ➡ **(dZ) (zCcXxVv)(TtGgBb)** | [A] **(JlLmMn)**

[B] ➡ [A] **(wAa) (SsFfD)** | [A] **(NoOpP)**

[A] ➡ **(RrQqE) (eW)** | **(hHilj)**

Análise Sintática – Árvore Sintática

Características

- Os nós intermediários serão formados por **símbolos não-terminais**.

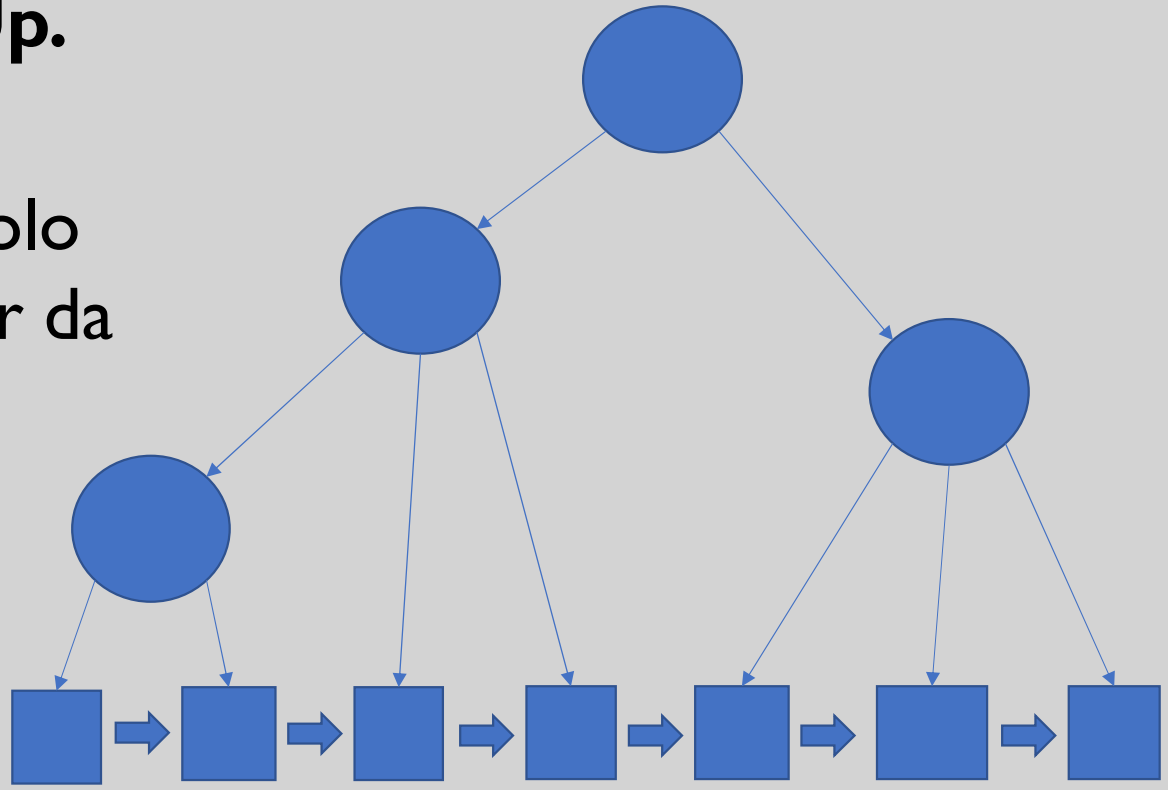
- As folhas serão formadas por **símbolos terminais**.

- A raiz da árvore será o **símbolo inicial** da gramática.

Análise Sintática - Ascendentes

Chamados de **Bottom-Up.**

Procuram chegar ao símbolo inicial da gramática a partir da sentença a ser analisada (Redução).



Análise Sintática – Bottom-Up

Exemplo - Validar a entrada **a+a*a** nas seguintes regras :

$$
\begin{aligned}
E &\rightarrow E + T \mid T \\
T &\rightarrow T * F \mid F \\
F &\rightarrow (E) \mid a
\end{aligned}
$$

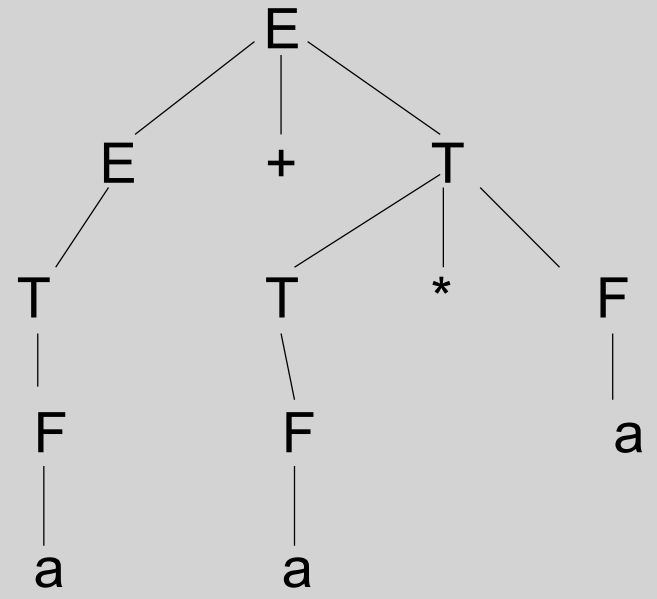
$$a+a*a \Leftarrow F+a*a \Leftarrow T+a*a \Leftarrow E+a*a \Leftarrow E+F*a \Leftarrow E+T*a \Leftarrow E+T*F \Leftarrow E+T \Leftarrow E$$

$$E*a \Leftarrow E*F \Leftarrow ?$$

Análise Sintática – Bottom-Up

Exemplo - Validar a entrada **a+a*a** nas seguintes regras :

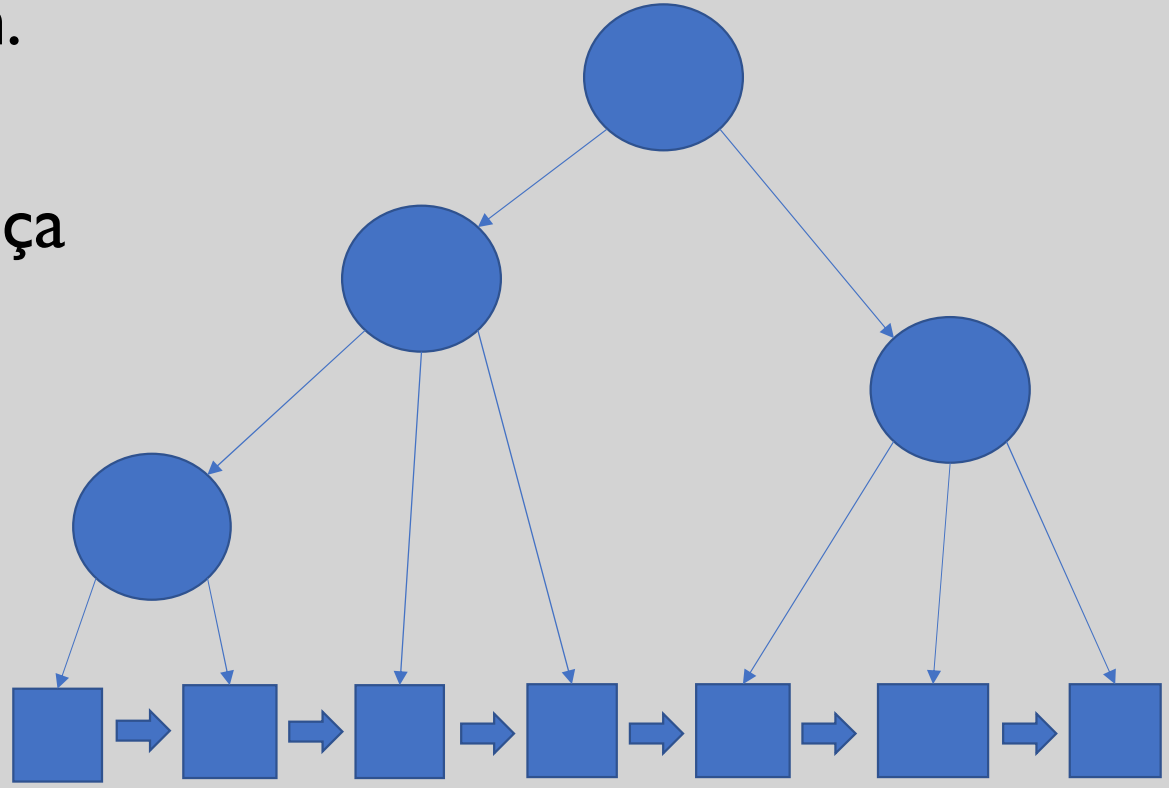
$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid a$$



Análise Sintática - Descendentes

Chamados de **Top-Down**.

Procuram chegar à sentença
a partir do símbolo inicial
da gramática (Derivação).



Análise Sintática – Top-Down

Exemplo - Chegar a sentença **a+a*a** nas seguintes regras :

$$
\begin{aligned}
E &\rightarrow E + T \mid T \\
T &\rightarrow T * F \mid F \\
F &\rightarrow (E) \mid a
\end{aligned}
$$

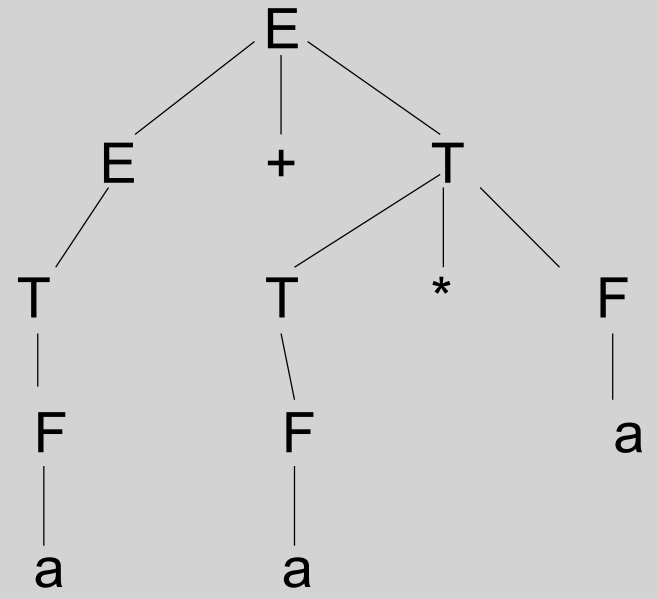
$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow a+T \Rightarrow a+T*F \Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow a+a*a$$

$$T \Rightarrow T*F \Rightarrow F*F \Rightarrow a*F$$

Análise Sintática – Top-Down

Exemplo - Chegar a sentença **a+a*a** nas seguintes regras :

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid a$$



Análise Sintática – Comparação

Ambos precisam tratar a possibilidade de voltar no processo de Derivação/Redução.

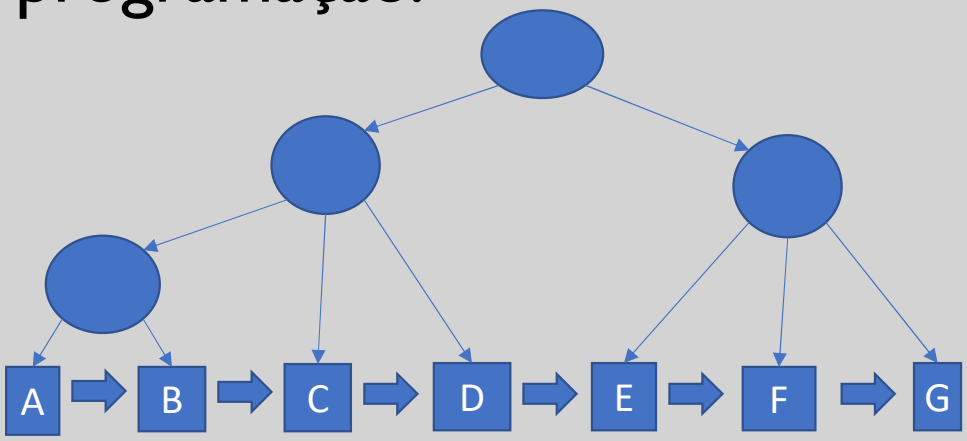
Bottom-Up resolve usando uma pilha auxiliar.

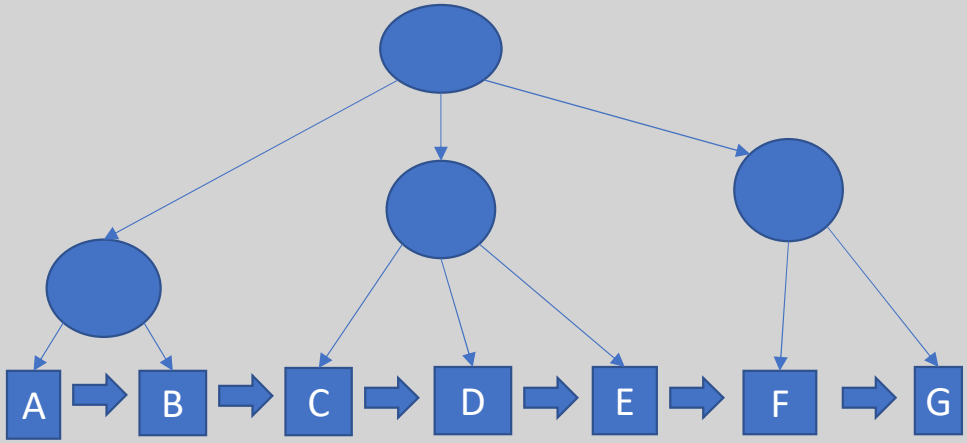
Top-Down consegue usar a pilha da recursão.

Análise Sintática - Ambiguidade

Uma gramática pode permitir mais de uma árvore sintática para mesma sequência de tokens.

Não é uma característica desejável para linguagem de programação.





Análise Sintática – Erros Sintáticos

Não encontrar uma regra que identifique uma sequência de tokens.

Podem ocorrer por:

- Ausência de token
- Inclusão indevida de token
- Troca de token (ausência + inclusão indevida)

Análise Sintática – Exemplos

$$
\begin{aligned}
E &\rightarrow E + T \mid T \\
T &\rightarrow T * F \mid F \\
F &\rightarrow (E) \mid a
\end{aligned}
$$

a + a * a - Entrada válida

Ausência: a a * a

Inclusão indevida: a + + a * a

Troca: a) a * a

Análise Sintática - Exemplo

```
program teste;
var
        a, b : real;
function inc( e : real) : real;
begin
        inc := e;
        inc := inc + 1
end
begin
        b := inc(a);
end;
```

Análise Sintática - Exemplo

[PGR] ⇒ **(program)** [ID] **(;)** [CORPO]

[CORPO] ⇒ [DECL] **(begin)** [COM] **(end)** |
(begin) [COM] **(end)**

[DECL] ⇒ [DCONST] [DTIPO] [DVAR] [DROT]

[DVARS] ⇒ **(var)** [VARS] | ε

[DROTS] ⇒ [FUNC] [DROTS] | ε

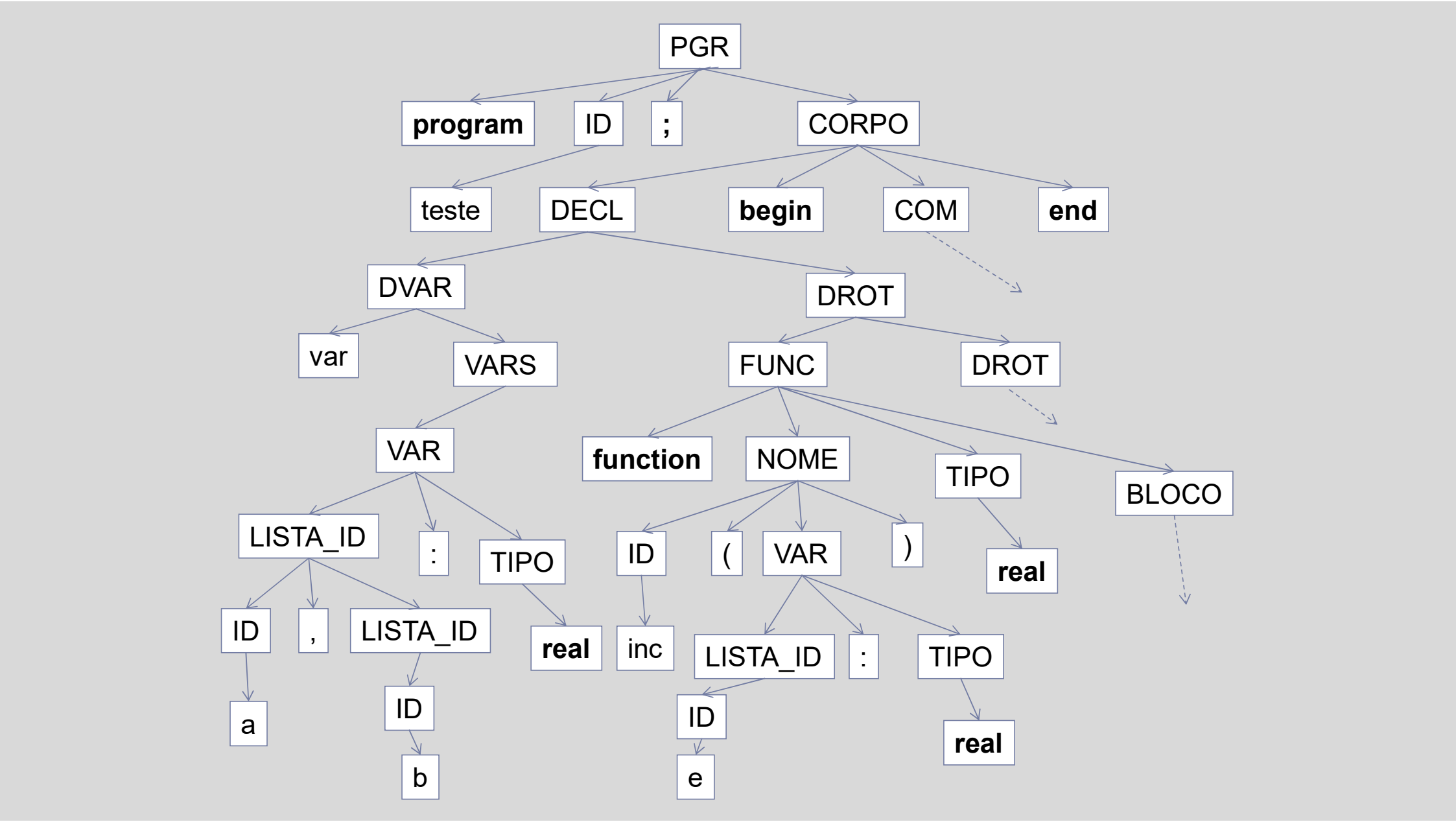
[VARS] ⇒ [VAR] **(;)** [VARS] | [VAR]

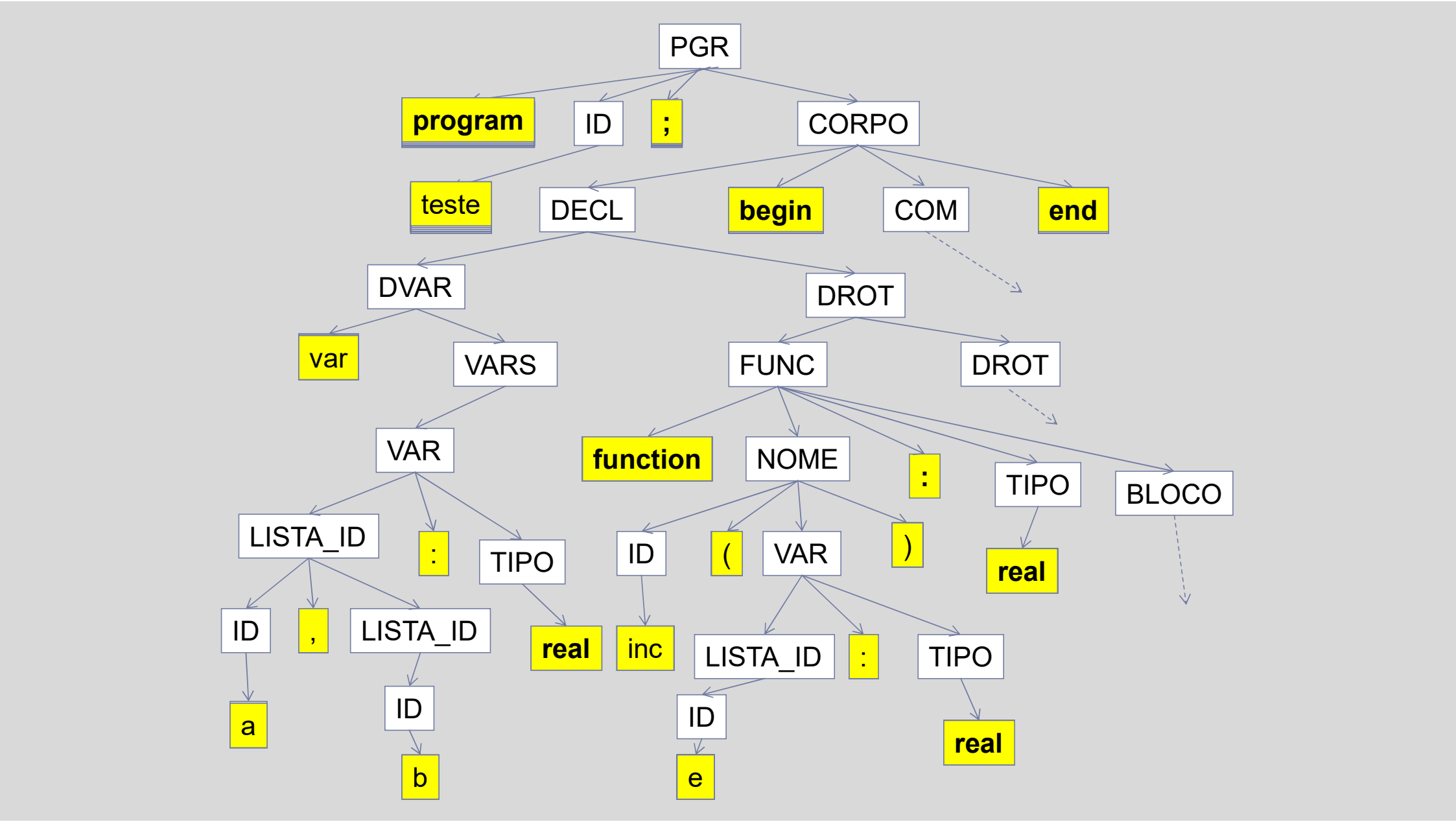
[VAR] ⇒ [LISTA_ID] **(:)** [TIPO]

[FUNC] ⇒ **(function)** [NOME] **(:)** [TIPO] [BLOCO]

Análise Sintática - Exemplo

program	- PROGRAMA
teste	- ID
;	- PT_VIRGULA
var	- VAR
a	- ID
,	- VIRGULA
b	- ID
:	- DOIS_PONTOS
real	- TIPO_DADO;
function	- FUNÇÃO
inc	- ID
(	- A_PARENTESE
e	- ID
:	- DOIS_PONTOS
real	- TIPO_DADO
)	- F_PARENTESE
:	- DOIS_PONTOS
real	- TIPO_DADO
;	- PT_VIRGULA
begin	- INICIO
inc	- ID
:=	- ATRIBUIÇÃO
e	- ID
;	- PT_VIRGULA
inc	- ID
:=	- ATRIBUIÇÃO
Inc	- ID
+	- OP_MAT
I	- NUMERAL
end	- FIM
begin	- INICIO
b	- ID
:=	- ATRIBUIÇÃO
inc	- ID
(	- A_PARENTESE
a	- ID
)	- F_PARENTESE
;	- PT_VIRGULA
end	- FIM
;	- PT_VIRGULA


PGR
program
ID
;
CORPO
teste
DECL
begin
COM
end
DVAR
DROT
var
VARS
FUNC
DROT
VAR
function
NOME
TIPO
BLOCO
LISTA_ID
:
TIPO
ID
(
VAR
)
real
ID
,
LISTA_ID
real
inc
LISTA_ID
:
TIPO
a
ID
ID
real
b
e


PGR
program
ID
;
CORPO
teste
DECL
begin
COM
end
DVAR
DROT
var
VARS
FUNC
DROT
VAR
function
NOME
:
TIPO
BLOCO
LISTA_ID
:
TIPO
ID
(
VAR
)
real
ID
,
LISTA_ID
real
inc
LISTA_ID
:
TIPO
a
ID
ID
real
b
e