

# Linguagens e Gramáticas

Linguagens Formais  
Hierarquia de Chomsky

Já vimos que

**Linguagem** é um conjunto de *cadeias* de símbolos sobre um alfabeto/vocabulário,  $V$ . É um subconjunto específico de  $V^*$ . Estas cadeias são denominadas ***sentenças da linguagem***, e são formadas pela justaposição de elementos individuais, os símbolos da linguagem.

Pode-se representar uma linguagem:

(1) Por meio da descrição do conjunto finito ou infinito de cadeias (Formalismo Descritivo)

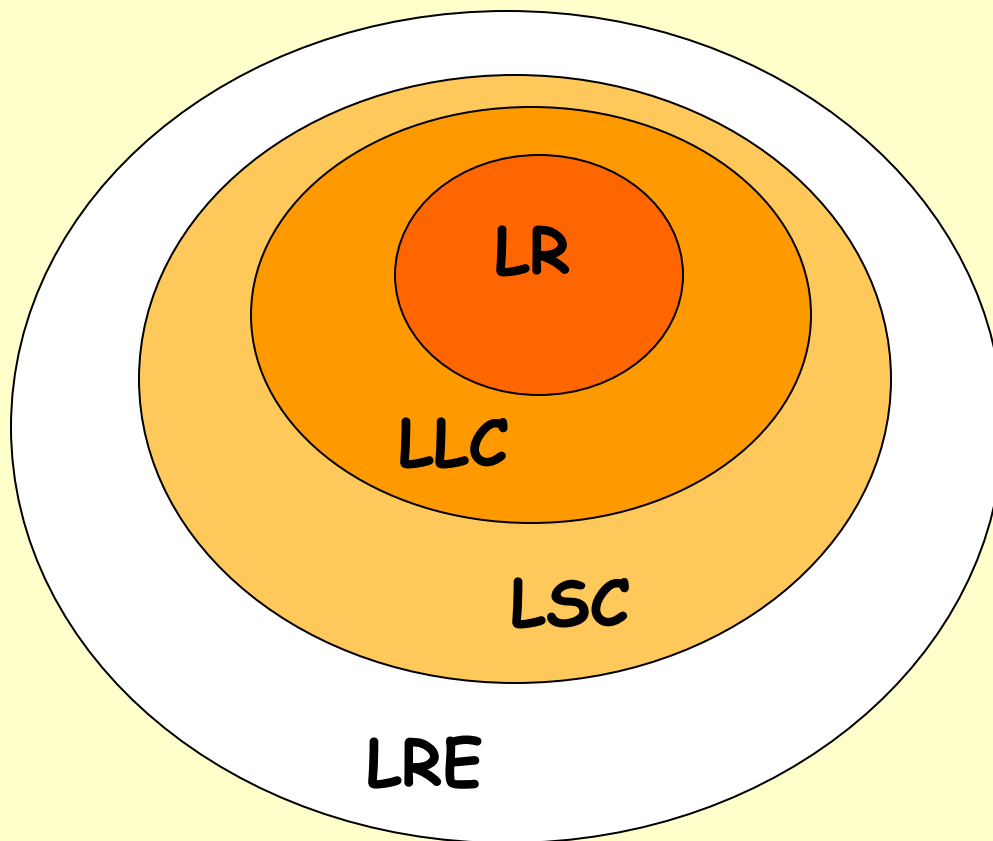
Ex: Linguagem dos números pares;  $L = \{ 0^n 1^n \mid n \geq 1 \}$

(2) por meio de um Autômato (Formalismo Reconhecedor - Autômato)

Ex: Máquina de Turing

(3) por meio de uma Gramática que a gere (Formalismo Gerativo)

## Hierarquia de Chomsky



**LR** = Linguagens Regulares

**LLC** = Linguagens Livres de Contexto

**LSL** = Linguagens Sensíveis ao Contexto

**LRE** = Linguagens Recursivamente Enumeráveis

# Tipos de Linguagens e Gramáticas

Linguagens	Gramáticas
Rec. Enumerável (LRE)	Estrutura de Frase ou ou Irrestrita ou do Tipo 0 (GEF)
Sensível ao contexto (LSC)	Sensível ao contexto ou do Tipo 1 (GSC)
Livre de contexto (LLC)	Livre de contexto ou do Tipo 2 (GLC)
Regular (LR)	Regular ou do Tipo 3 (GR)

# Tipos de Gramáticas

- O tipo mais geral de gramática é conhecido como **tipo 0** ou com **Estrutura de Frase ou Irrestritas**

$$\alpha \rightarrow \beta \quad \alpha \in V^+; \beta \in V^*$$

- Não há restrições nas regras de produção, a menos de que a cadeia nula não pode ocorrer do lado esquerdo.

$$G_1 = ( \{S, A, B, C, D, E\}, \{a\}, P, S )$$

$$P = \{ 1. S \rightarrow ACaB$$

$$2. Ca \rightarrow aaC$$

$$3. CB \rightarrow DB$$

$$4. CB \rightarrow E$$

$$5. aD \rightarrow Da$$

$$6. AD \rightarrow AC$$

$$7. aE \rightarrow Ea$$

$$8. AE \rightarrow \lambda \} \quad L(G_1) = ?$$

- Menor cadeia: aa

$$S \Rightarrow^1 ACaB \Rightarrow^2 AaaCB \Rightarrow^4 AaaE \Rightarrow^7 AaEa \Rightarrow^7 AEaa \Rightarrow^8 aa$$

- **A e B servem como marcadores da esq e dir para as formas sentenciais.**
- **C é o marcador que se move através da cadeia de a's entre A e B, dobrando seu número pela produção 2.**
- **Quando C alcança o marcador à direita B, ele se torna um D ou E pela produção 3 ou 4.**
- **Se um D é escolhido, então ele migra à esquerda pela produção 5 até que o marcador à esq, A, seja alcançado.**
- **Nesse ponto, D se torna C de novo pela produção 6 e o processo recomeça.**
- **Se um E é escolhido, o marcador à direita (B) é consumido.**
- **O E migra à esquerda pela produção 7 e consome o marcador à esq pela produção 8.**

$$L(G1) = \{a^{2^n} \mid n \text{ é um inteiro positivo}\}$$

# Classes Gramaticais

Conforme as restrições impostas ao formato das produções de uma gramática, a classe de linguagens que tal gramática gera varia correspondentemente. A teoria mostra que há quatro classes de gramáticas capazes de gerar quatro classes correspondentes de linguagens, de acordo com a denominada Hierarquia de Chomsky:

Gramáticas com Estrutura de Frase ou Irrestritas ou Tipo 0

Gramáticas Sensíveis ao Contexto ou Tipo 1

Gramáticas Livres de Contexto ou Tipo 2

Gramáticas Regulares ou Tipo 3



# Linguagens LRE

As linguagens geradas pelas Gramáticas com Estrutura de Frase ou do Tipo 0 são chamadas de Linguagens Recursivamente Enumeráveis (LRE) ou Linguagens do Tipo 0.

São as linguagens para as quais há uma Máquina de Turing que as reconhece.

Correspondem às funções computáveis. <sup>9</sup>

# Gramáticas Sensíveis ao/Dependentes de Contexto ou Tipo 1

Se às regras de substituição for imposta a restrição de que nenhuma substituição possa reduzir o comprimento da forma sentencial à qual a substituição é aplicada, cria-se uma classe de gramáticas ditas sensíveis ao contexto.

As gramáticas que obedecem a estas restrições pertencem, na hierarquia de Chomsky, ao conjunto das **Gramáticas Sensíveis ao Contexto (GSC) ou do Tipo 1**.

Para as GSC, as produções são todas da forma

$$\alpha \rightarrow \beta, \text{ com } |\alpha| \leq |\beta|$$

$$\text{onde } \alpha, \beta \in (V_n \cup V_t)^+$$

(produções não decrescentes)

Alguns autores colocam as produções de uma GSC como:

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2 \text{ com } \alpha_1, \alpha_2, \beta \in V^*, \beta \neq \lambda \text{ e } A \in V_n$$

para motivar o nome "sensível ao contexto" desde que a produção  $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$  permite que  $A$  seja trocado por  $\beta$  no contexto de  $\alpha_1$  e  $\alpha_2$ .

$$G_2 = (\{A, B, C\}, \{a, b, c\}, P, A)$$

$$P = \left\{ \begin{array}{ll} A \rightarrow abc & A \rightarrow aBbc \\ Bb \rightarrow bB & Bc \rightarrow Cbcc \\ bC \rightarrow Cb & aC \rightarrow aaB \\ aC \rightarrow aa \end{array} \right\}$$

$$L(G_2) = a^n b^n c^n, n \geq 1$$

# Linguagens Sensíveis ao Contexto - LSC

As linguagens geradas pelas Gramáticas Sensíveis ao Contexto ou do Tipo 1 são chamadas de Linguagens Sensíveis ao Contexto (LSC) ou Linguagens do Tipo 1.

## Resultado 1:

Toda gramática do tipo 1 é também do tipo 0.

## Corolário 1:

Toda LSC é também uma LRE (mas nem toda LRE é LSC).

# Gramáticas Livres de Contexto ou Tipo 2

As *Gramáticas Livres de Contexto (GLC)* ou do Tipo 2 são aquelas cujas regras de produção são da forma:

$$A \rightarrow \alpha \quad \text{onde } A \in V_n, \alpha \in V^+$$

Ou seja, quando do lado esquerdo da regra há apenas um símbolo não-terminal

A gramática  $G_3$  é uma GLC.

$$G_3 = (\{S, A, B\}, \{a, b\}, P, S)$$

$$P = \{S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB \}$$

$$L(G_3) = ?$$

- Menores cadeias: ab e ba

$S \Rightarrow aB \Rightarrow ab$

$S \Rightarrow bA \Rightarrow ba$

$S \Rightarrow aB \Rightarrow abS \Rightarrow abbA \Rightarrow abba$

$\Rightarrow abaB \Rightarrow abab$

$\Rightarrow aaBB \Rightarrow aabb$

$S \Rightarrow bA \Rightarrow baS \Rightarrow baaB \Rightarrow baab$

$\Rightarrow babA \Rightarrow baba$

$\Rightarrow bbAA \Rightarrow bbaa$

$L(G3) = \{w \in \{a,b\}^+ \mid nro(a) = nro(b)\}$

Todas as combinações de cadeias em  $V^+$  com  $nro(a) = nro(b)$

# Linguagens Livres de Contexto (LLC)

As linguagens geradas pelas Gramáticas Livres de Contexto ou do Tipo 2 são chamadas de Linguagens Livres de Contexto (LLC) ou Linguagens do Tipo 2.

## Resultado 2:

Toda gramática do tipo 2 é também do tipo 1.

## Corolário 2:

Toda LLC é também uma LSC (mas nem toda LSC é uma LLC).

## Corolário 2.1:

Toda LLC é também uma LRE (mas nem toda LRE é uma LLC)

# BNF

Outra maneira de se representar as Gramáticas Livres de Contexto é através da **Forma Normal de Backus**.

Neste caso,  $\rightarrow$  é substituído por  $::=$  e os não terminais são ladeados por  $\langle \rangle$

No caso de repetições de lado esquerdo:

$\langle A \rangle ::= a_1$

$\langle A \rangle ::= a_2$

:

$\langle A \rangle ::= a_n$

escreve-se:  $\langle A \rangle ::= a_1 | a_2 | \dots | a_n$

Os símbolos  $\langle, \rangle$ ,  $::=$ ,  $|$  formam a metalinguagem, ou seja, são símbolos que não fazem parte da linguagem mas ajudam a descrevê-la.



Exemplo:

$G4 = \{Vn, Vt, P, S\}$  onde:

$Vn = \{\langle \text{sentença} \rangle, \langle \text{sn} \rangle, \langle \text{sv} \rangle, \langle \text{artigo} \rangle, \langle \text{substantivo} \rangle, \langle \text{verbo} \rangle\}$

$Vt = \{o, a, \text{peixe}, \text{comeu}, \text{isca}\}$

$S = \langle \text{sentença} \rangle$

$P = \{$

1.  $\langle \text{sentença} \rangle ::= \langle \text{sn} \rangle \langle \text{sv} \rangle$

2.  $\langle \text{sn} \rangle ::= \langle \text{artigo} \rangle \langle \text{substantivo} \rangle$

3.  $\langle \text{sv} \rangle ::= \langle \text{verbo} \rangle \langle \text{sn} \rangle$

4.  $\langle \text{artigo} \rangle ::= o|a$

5.  $\langle \text{verbo} \rangle ::= \text{mordeu}$

6.  $\langle \text{substantivo} \rangle ::= \text{peixe}|isca \}$

Exercícios:

- verifique se a cadeia "a isca mordeu o peixe" é uma sentença de  $L(G4)$ .
- Dê exemplos de sentenças de  $L(G4)$ .

## Mais GLC:

$G5 = (\{S\}, \{a, +, *, (, )\}, P, S)$

$P = \{$   
     $S \rightarrow S * S$   
     $S \rightarrow S + S$   
     $S \rightarrow (S)$   
     $S \rightarrow a \}$

$L(G5) =$  conjunto das expressões aritméticas envolvendo  
 $*, +, (, )$  e  $a$ .

Um exemplo de cadeia formada por esta gramática é  
 $a * (a + a)$ .

Processo inverso: Dada uma  $L(G)$  definir a gramática  $G$ .

$$L(G) = \{a^m b^n \mid m \geq 1, n \geq 1\}$$

$L(G_6) = \{a^m b^n \mid m \geq 1, n \geq 1\}$  ou  $a^+ b^+$

Resp.:

$G_6 = (\{S, A, B\}, \{a, b\}, P, S)$

$P = \{S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b\}$

*Obs. : Caso geral:*

*Se  $A \rightarrow \alpha A \mid \beta$  então  $A \Rightarrow \alpha^* \beta$*

$$L(G7) = \{a^n b^n \mid n \geq 1\}$$

$$L(G) = \{a^n b^n \mid n \geq 1\}$$

$$G = (\{S\}, \{a, b\}, P, S)$$

$$P = \{S \rightarrow aSb \mid ab\}$$

# Árvores de Derivação Sintática

- Em uma gramática é possível haver várias derivações equivalentes (usam as mesmas produções nos mesmos lugares MAS em ordem diferente).
- Para GLC temos uma representação gráfica que representa uma classe de equivalências chamada **Árvore de Derivação**. Ela deve ser única para uma sentença.
- Uma **árvore de derivação** para uma sentença gerada por uma GLC  $G = (V_n, V_t, P, S)$  é uma **árvore rotulada ordenada** em que cada nó é rotulado por um símbolo de  $V_n \cup V_t \cup \lambda$ .
- Se um nó interno é rotulado com  $A$  e seus descendentes diretos são rotulados com  $X_1, X_2, \dots, X_n$  então  
 $A \rightarrow X_1 X_2 \dots X_n$  é uma produção de  $P$ .

Uma árvore rotulada ordenada  $D$  é uma árvore de derivação para uma GLC  $G(A) = (V_n, V_t, P, A)$  se

- (1) A raiz de  $D$  é rotulada com  $A$  (o axioma)
- (2) Se  $D_1, \dots, D_k$  são as subárvores de descendentes diretos da raiz e a raiz de  $D_i$  é rotulada com  $X_i$ ,  
então  $A \Rightarrow X_1, \dots, X_k$  é uma produção em  $P$ .  
 $D_i$  deve ser a árvore de derivação para  $G(X_i) = (V_n, V_t, P, X_i)$  se  $X_i$  é não-terminal, e  $D_i$  é um nó simples rotulado com  $X_i$  se  $X_i$  é terminal.
- (3) Se  $D_1$  é a única subárvore da raiz  $D$  e a raiz de  $D_1$  é rotulada com  $\lambda$  então  $A \rightarrow \lambda$  é uma produção de  $P$ .

Vértices internos são não-terminais e vértices folhas podem ser não-terminais, terminais ou  $\lambda$ .

Quando fazemos a árvore de derivação de uma sentença, os vértices folhas são sempre terminais ou  $\lambda$ .

Uma sentença está representada na árvore de derivação fazendo-se a leitura das folhas (nós sem descendentes) da esquerda para direita

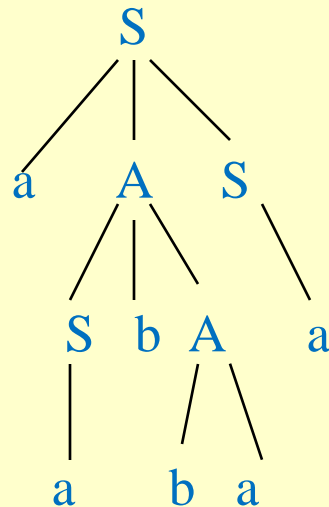


## Exemplos

- Árvore de derivação para a sentença **aabbaa** da GLC  $G = (\{S,A\}, \{a,b\}, P, S)$

P:  $S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid ba \mid SS$



- Seja  $G_a = (\{E, T, F\}, \{+, *, (, ), a\}, P, E)$

$$P: \quad E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

1) Seja a sentença  $a + a * a$ . Mostre a derivação mais à esquerda e a mais à direita.

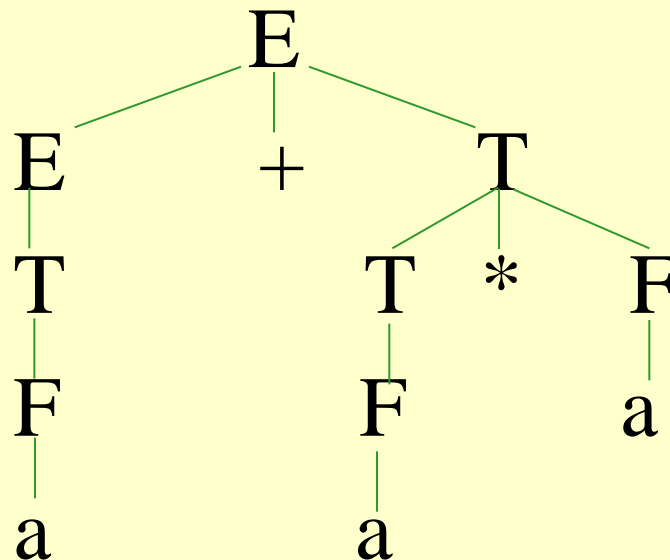
- Derivação mais à esquerda (derivar primeiro o não-terminal mais à esquerda):

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

- Derivação mais à direita (derivar primeiro o não-terminal mais à direita):

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

2) Agora, mostre a árvore de derivação para  $a + a * a$



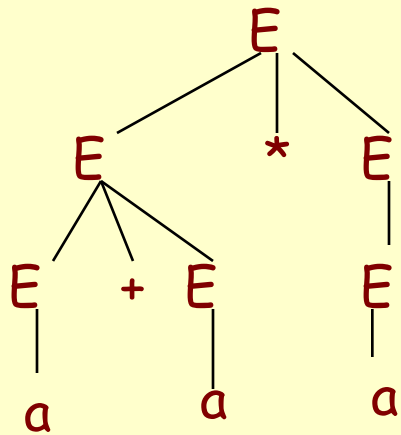
Única Árvore de Derivação

Agora considere a gramática Gb:

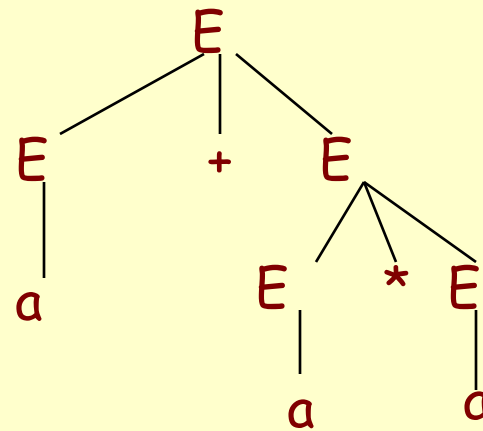
$$E \rightarrow E + E \mid E * E \mid (E) \mid a$$

Não é difícil ver que  $L(Ga) = L(Gb)$ .

Faça a árvore de derivação mais à esquerda de  $a+a^*a$  para Gb:



Não é  
única!!



$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$

Der.  
esq.

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$

Der.  
esq.

→ Gramática Ambígua

## De fato:

- Ga indica que o operador  $*$  tem precedência sobre o operador  $+$
- Por Gb, ambos operadores têm igual precedência.
- A linguagem de uma gramática ambígua é dita ambígua.
- A ambiguidade decorre do fato de que as árvores de derivação implicam interpretações distintas.

# Ambiguidade e Derivações mais à Esquerda

Teorema: Para cada gramática

$G=(V_n, V_t, P, S)$  e cadeias  $w$  em  $V_t^*$ ,  $w$  tem duas árvores de análise sintática distintas (*logo,  $G$  é ambígua*) se e somente se  $w$  tem duas derivações mais à esquerda a partir de  $S$ .

# Ambiguidade nas GLC e as Linguagens de Programação (LP)

- Um requisito importante de uma LP é que ela **não seja ambígua**.
- O mais famoso caso de ambiguidade é o **else pendente**, presente na especificação de muitas LP.

- Seja a gramática:

$C \rightarrow \text{if } b \text{ then } C \text{ else } C$

$C \rightarrow \text{if } b \text{ then } C$

$C \rightarrow s$

Ela é ambígua desde que a cadeia

$\text{If } b \text{ then if } b \text{ then } s \text{ else } s$

Pode ser interpretada como

(i) If b then (if b then s else s)

S : outro comando qualquer

Ou

(ii) If b then (if b then s) else s

A primeira é a preferida nas LP pois utiliza a regra informal “**case o else com o if mais próximo**”, que resolve a ambiguidade



- Para eliminar a ambiguidade da gramática anterior, podemos reescrevê-la com 2 não-terminais C1 e C2:

C1 → if b then C1 | if b then C2 else C1 | s

C2 → if b then C2 else C2 | s

S : outro comando qualquer

O fato de que **somente C2 precede o else** garante que, entre o par **then-else** gerado por qualquer uma das produções, deve aparecer ou um **s** ou outro **then-else**. Assim, a interpretação (ii) nunca ocorre.

(ii) If b then (if b then s) else s

Obs.: se essa for a interpretação desejada, então, nas LPs, usamos begin-end.

# Como mostrar que uma gramática é ambígua?

- Com árvores de derivação.
- **Def1:** Uma GLC (G) é ambígua se há pelo menos uma cadeia pertencente à  $L(G)$  com mais de uma árvore de derivação para representá-la.
- **Def2:** A existência de uma sentença com duas ou mais derivações mais à esq (ou mais à dir) caracteriza uma linguagem ambígua.

# Gramática de Operadores

- Para retirar a ambiguidade de gramáticas de operadores:
  - introduzimos várias variáveis e estratificamos as regras, quando temos operadores com várias prioridades de resolução.
  - Para resolver a ambiguidade vinda do uso de vários operadores idênticos, forçamos o uso da recursão para esquerda ou direita (associatividade).

# Regras de Prioridade e Associatividade

- Ajudam na decisão da interpretação correta de expressões nas LP
- **Def:** Um operador é **associativo à esquerda** se os operandos são agrupados da esq para dir, e é **associativo à direita** se os operandos são agrupados da dir para esq.

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

+ e\*: associativos à esquerda

Ex.:  $a + a + a$  (**1º.**; **2º.**)

- Está relacionada com a posição da recursão nas regras que permitem um operador ser aplicado mais de uma vez:
- $E \rightarrow E + T \rightarrow E + T + T \rightarrow T + T + T \rightarrow \dots \rightarrow a+a+a$

- Os níveis de **prioridade** indicam a quais operadores é permitido agrupar seus operandos primeiro (resolver primeiro).

not

Prioridade

\* / div mod and

+ - or

< > <> >= <= =

Quanto mais distante do axioma estiver o operador, maior sua prioridade.

# Linguagens Inerentemente ambíguas

É simples encontrar um exemplo de GLC ambígua. Na gramática abaixo para a sentença “a” temos 2 árvores

$$S \rightarrow A \mid B$$
$$A \rightarrow a$$
$$B \rightarrow a$$

Mas não é tão simples exibir uma LLC inerentemente ambígua

Def 😞: Uma LLC é inerentemente ambígua se não há uma gramática não-ambígua que a gere.

$$L = \{a^i b^j c^k \mid i, j, k \geq 1 \text{ e } i = j \text{ OU } j = k\}$$

$S \rightarrow abc \mid aRbI \mid YbWc$

$I \rightarrow Ic \mid c$

$R \rightarrow ab \mid aRb$

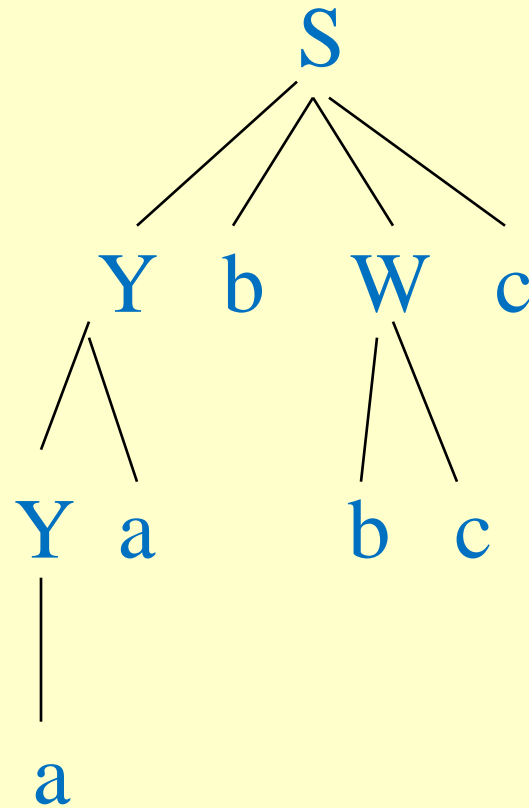
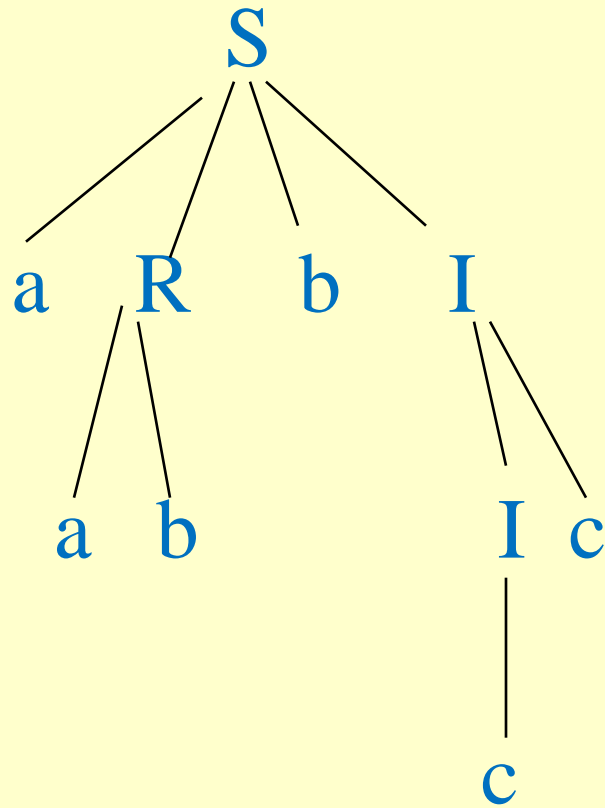
$Y \rightarrow Ya \mid a$

$W \rightarrow bc \mid bWc$

O fato de ser inerentemente ambígua decorre de que toda gramática que gera  $L$  gera cadeias para  $i=j$  por um processo diferente do qual usa para gerar as cadeias para  $j=k$ .

É impossível não gerar algumas cadeias para as quais  $i=j=k$  por ambos os processos.  **$L$  é ambígua.**

- Exemplo: aabbcc





# Propriedades de Gramáticas

## Ambíguas

Teo ☹️: Não existe um algoritmo tal que, dada uma GLC qualquer, retorne a resposta sim, se ela for ambígua, ou não, se ela não for ambígua. (Logo, essa propriedade é indecidível)

MAS

- ☺️ em casos particulares, nós podemos reconhecer a ambiguidade e remove-la “à mão”
- ☺️ E podemos utilizar classes mais restritas de GLC que, por definição, não são ambíguas (ex. LL(K))

# Exemplos de produções ambíguas

1.  $A \rightarrow AA$
2.  $A \rightarrow A \alpha A$
3.  $A \rightarrow \alpha A \mid A\beta$
4.  $A \rightarrow \alpha A \mid \alpha A\beta A$

# Quais Gramáticas são ambíguas?

(1)  $S \rightarrow bA \mid aB$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

(2) A gramática usada para gerar expressões aritméticas na notação posfix na linguagem APL:

$S \rightarrow SS + \mid SS - \mid SS * \mid x \mid y$

ou na notação prefix:

$S \rightarrow +SS \mid -SS \mid *SS \mid x \mid y$

(3) A gramática para gerar parênteses aninhados:

$S \rightarrow (S) \mid ( ) \mid SS$

(4) A gramática que define os operadores lógicos and e or

$E \rightarrow E \text{ or } E \mid E \text{ and } E \mid (E) \mid a$

Classifique as gramáticas (GEF, GSC, GLC), dê a quádrupla e a  $L(G)$  e diga se são finitas/infinitas ou ambíguas

$$1) \quad E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid F \\ F \rightarrow 0 \mid 1 \mid \dots \mid 9$$

$$2) \quad A \rightarrow BC \\ BC \rightarrow CB \\ B \rightarrow b \\ C \rightarrow a$$

$$3) \quad A \rightarrow 0A \mid B \\ B \rightarrow 1B \mid \lambda$$

$$4) S \rightarrow 0A$$
$$A \rightarrow 1S \mid 1$$

$$5) S \rightarrow 0A$$
$$A \rightarrow 1B$$
$$B \rightarrow 1S \mid 1$$

$$6) L(G) = \{111(00)^*\}$$
$$G = ?$$

$$7) L(G) = \{a^n b^n c^* \mid n \geq 1 \text{ e } i \geq 0\}$$
$$G = ?$$

$$8) L(G) = \{a^* b^n c^n \mid n \geq 1 \text{ e } i \geq 0\}$$
$$G = ?$$

9) Utilize o software JFLAP com os exemplos acima