

# 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) como um conjunto finito ou infinito de cadeias

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

(2) por meio de uma Máquina de Turing que a reconheça

(3) Por meio de uma Gramática que a gere

A Gramática é o formalismo **gerativo** de linguagens, enquanto que os automatos (a MT é um automato) são **reconhecedores** de linguagens.

## Exemplo 1:

$V_n$

$V_t$

Axioma

$G_1 = (\{S, A, B\}, \{a, b\}, P, S)$   
 $P = \{$   
  1.  $S \rightarrow aB$   
  2.  $S \rightarrow bA$   
  3.  $A \rightarrow a$   
  4.  $A \rightarrow aS$   
  5.  $A \rightarrow bAA$   
  6.  $B \rightarrow b$   
  7.  $B \rightarrow bS$   
  8.  $B \rightarrow aBB$   $\}$

P: Regras de  
Produção

Como  $G$  gera uma linguagem? Por um processo de substituição ou derivação de símbolos. As cadeias  $\in V_t^*$  geradas formam a linguagem  $L(G)$ .

$S \xrightarrow{1} aB \xrightarrow{8} aaBB \xrightarrow{7} aabB \xrightarrow{7} aabb$   
 $S \xrightarrow{2} bA \xrightarrow{4} baS \xrightarrow{1} baaB \xrightarrow{7} baab$   
etc.

## Definição:

Formalmente, as gramáticas são caracterizadas como quádruplas ordenadas

$$G = ( V_n, V_t, P, S )$$

onde:

1.  $V_n$  representa o vocabulário não terminal da gramática. Este vocabulário corresponde ao conjunto de todos os símbolos dos quais a gramática se vale para definir as leis de formação das sentenças da linguagem.

**2.  $V_t$**  é o vocabulário terminal, contendo os símbolos que constituem as sentenças da linguagem. Dá-se o nome de terminais aos elementos de  $V_t$ .

**3.  $P$**  representa o conjunto de todas as leis de formação utilizadas pela gramática para definir a linguagem.

Para tanto, cada construção parcial, representada por um não-terminal, é definida como um conjunto de regras de formação relativas à definição do não-terminal a ela referente. A cada uma destas regras de formação que compõem o conjunto  $P$  dá-se o nome de produção da gramática.

Assumimos  $V_n \cap V_t = \emptyset$ . Convencionamos que  $V_n \cup V_t = V$   
Cada produção  $P$  tem a forma:

$\alpha \rightarrow \beta$       $\alpha \in V^+$ ; (qualquer cadeia não nula de  $V$ ) e  
                   $\beta \in V^*$  (qualquer cadeia de  $V$ , incluindo a nula)

4.  $S \in V_n$  denota a principal categoria gramática de  $G$ ; é dito o símbolo inicial ou o axioma da gramática. Indica onde se inicia o processo de geração de sentenças.

# Notação/Convenções

- Letras do alfabeto latino maiúsculas {A,B,..Z}: **variáveis**
- Letras do começo do alfabeto latino minúsculas {a,b,c,...}: **terminais**
- Letras do fim do alfabeto latino minúsculas {t,u,v,x,z}: **cadeias de terminais**
- Letras gregas minúsculas { $\alpha, \beta, \gamma, \delta, \epsilon, \dots, \omega$ }: **cadeias de terminais e não terminais.**
- Regras de produção com mesmo lado esquerdo são simplificadas com a notação | **(ou)**

$$G1 = (\{S,A,B\}, \{a,b\}, P, S)$$

$$P1 = \{S \rightarrow aB \mid bA$$

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

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



Definida uma gramática  $G$ , qual é a linguagem gerada por ela?

Sejam as relações

$\Rightarrow$  (deriva/gera diretamente) e

$\Rightarrow^*$  (deriva/gera)

definidas entre as cadeias de  $V^*$

**Def.1.** Se  $\alpha \rightarrow \beta$  é uma produção de  $P$  e  $\gamma$  e  $\delta$  são cadeias quaisquer de  $V^*$ , então

$$\gamma \alpha \delta \Rightarrow \gamma \beta \delta$$

Ou:  $\alpha$  deriva/gera  $\beta$  ( $\beta$  é derivado de  $\alpha$ ) por uma única produção, não importa o contexto em que aparecem.

No Ex.1.:  $S \Rightarrow aB$  ;  $aB \Rightarrow aaBB$ ;  $B \Rightarrow b$       ou  
 $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabb$

**Def.2.** Suponha que  $\alpha_1 \alpha_2 \alpha_3 \dots \alpha_m$  são cadeias de  $V^*$  e

$$\alpha_1 \Rightarrow_G \alpha_2, \quad \alpha_2 \Rightarrow_G \alpha_3, \quad \dots, \quad \alpha_{m-1} \Rightarrow_G \alpha_m$$

Então dizemos que

$$\alpha_1 \Rightarrow^* \alpha_m$$

Ou:  $\alpha_1$  deriva/gera  $\alpha_m$  aplicando-se um número qualquer de produções de  $P$ .

Por convenção  $\alpha \Rightarrow^* \alpha$  para toda cadeia  $\alpha$ .

No Ex.1.:  $S \Rightarrow^* ab$ ;

$S \Rightarrow^* aaBB$ ;

$baS \Rightarrow^* baab$ ;

$aB \Rightarrow^* abbA$

**Def.3.** Toda cadeia derivada/gerada do símbolo inicial  $S$  é chamada uma **forma sentencial**.

Ou seja, uma cadeia  $\alpha \in V^*$  é uma forma sentencial se  $S \Rightarrow^* \alpha$

No Ex.1:  $aB$ ,  $AB$ ,  $S$ ,  $ab$  são formas sentenciais.

**Def.4.** Uma forma sentencial,  $\alpha$ , é uma **sentença** de  $G$  se for composta apenas de símbolos terminais.

Ou:  $\alpha \in V^*$  é uma sentença se

(a)  $S \Rightarrow^* \alpha$  (for forma sentencial) e

(b)  $\alpha \in Vt^*$  (só tem terminais).

Assim, as sentenças são as cadeias de terminais geradas pela gramática (por seu símbolo inicial).

**Def.5.** A Linguagem L gerada por uma gramática G é definida como o conjunto de sentenças de G. Ou seja,

$$L(G) = \{x \mid x \in Vt^* \text{ e } S \Rightarrow^* x\} \text{ ou } \{x \mid x \text{ é sentença de } G\}$$

1. A cadeia consiste somente de terminais
2. A cadeia é derivada a partir do símbolo inicial da gramática

**Def.6.** Duas gramáticas G1 e G2 são equivalentes sse  $L(G1) = L(G2)$

# Exemplos de Gramáticas

$G_2 = (\{S\}, \{0,1\}, P_1, S)$

$P: \{$   
1.  $S \rightarrow 0S1$   
2.  $S \rightarrow 01$   
 $\}$

Qual é a linguagem gerada por  $G_1$ ? Aplicamos o **processo de derivação** para saber  $L(G_1)$ , que é o processo de obtenção de cadeias a partir de uma gramática.

$G_3 = (\{S,B,C\}, \{a,b,c\}, P_2, S)$

$P: \{$   
1.  $S \rightarrow aSBC$   
2.  $S \rightarrow aBC$   
3.  $CB \rightarrow BC$   
4.  $aB \rightarrow ab$   
5.  $bB \rightarrow bb$   
6.  $bC \rightarrow bc$   
7.  $cC \rightarrow cc$   
 $\}$

$L(G_3) = ?$

## G2

- A menor cadeia gerada é 01:  $S \Rightarrow^2 01$
- Se aplicarmos  $n-1$  vezes a produção 1, seguida da produção 2 teremos:
  - $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0^3S1^3 \Rightarrow^*$
  - $0^{n-1}S1^{n-1} \Rightarrow 0^n1^n$
  - Portanto,  $L(G2) = \{0^n1^n \mid n \geq 1\}$   
ou  $S \Rightarrow^* 0^n1^n$

# G3

- A menor cadeia gerada é abc:  $S \Rightarrow^2 aBC \Rightarrow^4 abC \Rightarrow^6 abc$
- Usamos 1 n-1 vezes:  $S \Rightarrow^* a^{n-1}S(BC)^{n-1}$
- Usamos a 2 uma vez:  $S \Rightarrow^* a^n(BC)^n$
- A 3 permite trocar B com C para que B's precedam os C's
  - Para n = 2 aaB**C**BC  $\Rightarrow$  aaBBCC (usamos a regra 3 **1** vez)
  - Para n = 3 aaaB**C**BCBC  $\Rightarrow$  aaaBB**C**BC  $\Rightarrow$  aaaBB**C**CC  $\Rightarrow$  aaaBBBCCC (usamos a regra 3 **3** vezes)
  - Para n = 4 aaaaB**C**BCBCBC  $\Rightarrow$  aaaaBB**C**BCBC  $\Rightarrow$  aaaaBB**C**CCBC  $\Rightarrow$  aaaaBBB**C**CCBC  $\Rightarrow$  aaaaBBB**C**CCC  $\Rightarrow$  aaaaBBBBCCC (usamos a regra 3 **5** vezes);
  - Para n = 5 usamos a 3 **10** vezes.
- Assim  $S \Rightarrow^* a^nB^nC^n$
- Usamos a 4 uma vez:  $S \Rightarrow^* a^nbB^{n-1}C^n$
- Aplicamos a 5 n-1 vezes:  $S \Rightarrow^* a^nb^nC^n$
- Aplicamos a 6 uma vez:  $S \Rightarrow^* a^nb^ncC^{n-1}$
- Aplicamos a 7 n-1 vezes:  $S \Rightarrow^* a^nb^nc^n$

$$L(G3) = \{a^nb^nc^n \mid n \geq 1\}$$

# Tipos de Gramáticas

- Chamamos o tipo de gramática que definimos **de 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.

$$G_4 = ( \{S, A, B, C, D, E\}, \{a\}, P, S )$$

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

$$2. Ca \rightarrow aaC$$

$$3. CB \rightarrow DB$$

$$4. CB \rightarrow E$$

$$5. aD \rightarrow Da$$

$$6. AD \rightarrow AC$$

$$7. aE \rightarrow Ea$$

$$8. AE \rightarrow \lambda \} \quad L(G_4) = ?$$

- 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.**

$$L(G4) = \{a^{2^n} \mid n \text{ é um inteiro positivo}\}$$

## 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 Recursivamente Enumeráveis 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.

# 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  $\alpha_2$ .

$G_4$  é GSC;  $G_3$  e suas equivalentes abaixo também são GSC:

$$G_5 = (\{A, B, C\}, \{a, b, c\}, P, A)$$

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

$$G_6 = (\{S, C\}, \{a, b, c\}, P, S)$$

$$P = \left\{ \begin{array}{l} S \rightarrow abc \\ ab \rightarrow aabbC \\ Cb \rightarrow bC \\ Cc \rightarrow cc \end{array} \right\}$$

# 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 LEF (mas nem toda LEF 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  $G_1$ , do Ex 1, é uma GLC.

$$G_1 = (\{S, A, B\}, \{a, b\}, P, S)$$

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

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

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

$$L(G_1) = ?$$

- Menores cadeias: ab e ba

$S \Rightarrow aB \Rightarrow ab$

$S \Rightarrow bA \Rightarrow ba$

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

$\Rightarrow abaB \Rightarrow abab$

$\Rightarrow aaBB \Rightarrow aabb$

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

$\Rightarrow babA \Rightarrow baba$

$\Rightarrow bbAA \Rightarrow bbaa$

$L(G1) = \{w \in \{a,b\}^+ \mid nro(a) = nro(b)\}$

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).

# 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:

$G7 = \{Vn, Vt, P, S\}$  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.  $\langle \text{substantivo} \rangle ::= \text{peixe}| \text{isca} \}$

Exercícios:

- verifique se a cadeia "a isca mordeu o peixe" é uma sentença de  $L(G7)$ .
- Dê exemplos de sentenças de  $L(G7)$ .

## Mais GLC:

$G_8 = (\{S\}, \{a, +, *, (, )\}, P, S)$

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

$L(G_8) =$  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  $G$ .

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

$L(G_9) = \{a^m b^n \mid m \geq 1, n \geq 1\}$  ou  $a^+ b^+$

Resp.:

$G_9 = (\{S, A, B\}, \{a, b\}, P, S)$

$P = \{S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b\}$

*Obs. : Caso geral:*

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

$$L(G10) = \{a^n b^n \mid n \geq 1\}$$

$$L(G10) = \{a^n b^n \mid n \geq 1\}$$

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

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



# 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**.
- Uma **árvore de derivação** para uma GLC  $G = (V_n, V_t, P, S)$  é uma **árvore rotulada ordenada** em que cada nó é rotulado por um símbolo de  $V_n \cup V_t \cup \epsilon$ .
- 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  $A$
- (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  $A \rightarrow \epsilon$  é uma produção de  $P$ .

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

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

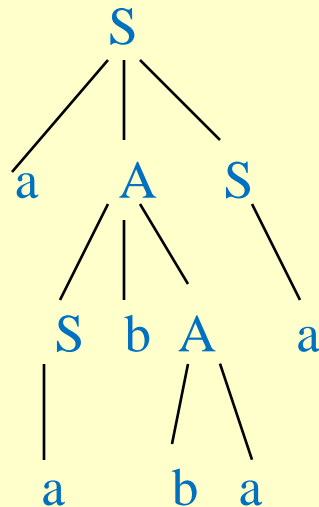
Uma sentença está representada na árvore de derivação fazendo-se a leitura das folhas (nós sem descendentes) da esquerda para direita

## 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.

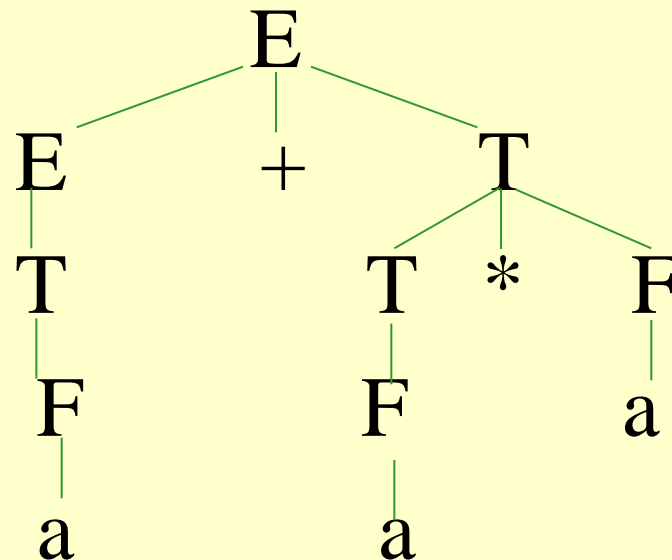
- Derivação mais à esquerda:

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

- Derivação mais à direita:

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

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



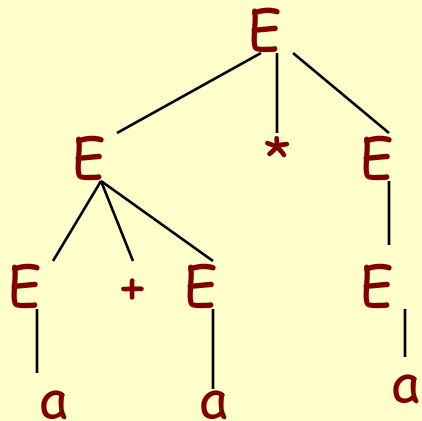
Única Árvore de Derivação

Agora considere a gramática  $G_b$ :

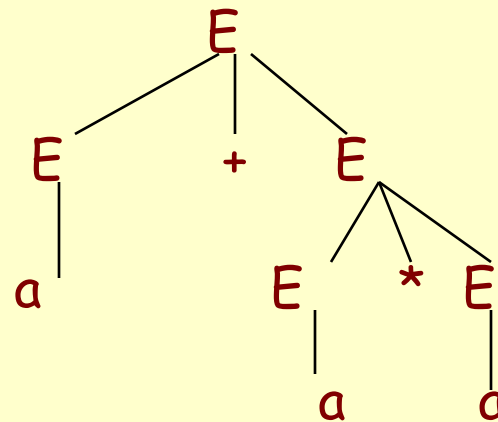
$$1. E \rightarrow E + E \mid E * E \mid (E) \mid a$$

Não é difícil ver que  $L(G_a) = L(G_b)$ .

Faça a árvore de derivação de  $a+a*a$  para  $G_b$ :



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

- 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 -> if b then C else C

C -> if b then C

C -> s

Ela é ambígua desde que a cadeia

If b then if b then s 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 em 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 reescreve-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 qq 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 Precedência, 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).

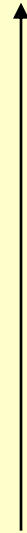
not

Prioridade

\* / div mod and

+ - or

< > <> >= <= =



# 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 toda gramática que a gera é ambígua



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

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

$I \rightarrow Ic \mid c$

$R \rightarrow ab \mid aRb$

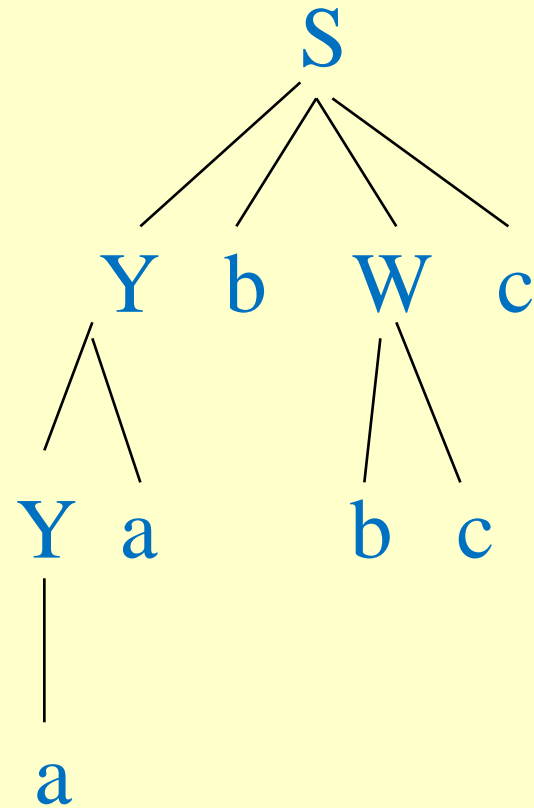
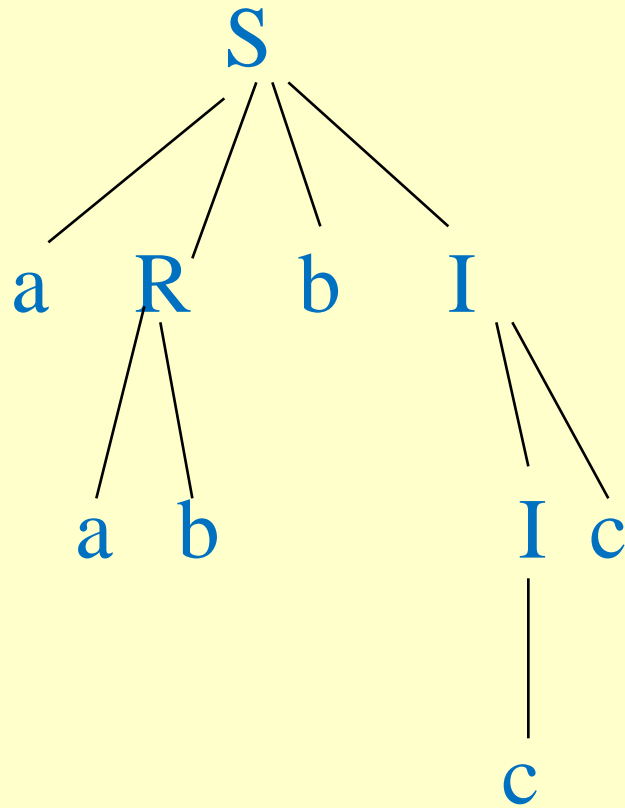
$Y \rightarrow Ya \mid a$

$W \rightarrow bc \mid bWc$

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.

- 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.

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))

Este problema (decidir se uma GLC é ambígua) é **parcialmente decidível**, pois, para determinadas gramáticas, o procedimento pára e diz sim, MAS pode não parar para outros casos.

# 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 na linguagem APL:

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

ou na notação :

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

(3) A gramática para gerar parênteses aninhados:

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

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

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

# Gramáticas Regulares ou Tipo 3

Aplicando-se mais uma restrição sobre a forma das produções, pode-se criar uma nova classe de gramáticas, as *Gramáticas Regulares (GR)*. Nas GRs, as produções são restritas às formas seguintes:

$A \rightarrow aB$  ou  $A \rightarrow a$  (linear à direita)

ou

$A \rightarrow Ba$  ou  $A \rightarrow a$  (linear à esquerda)

onde  $A, B \in V_n$  e  $a \in V_t$

# Linguagens Regulares (LR)

As linguagens geradas pelas Gramáticas Regulares ou do Tipo 3 são chamadas de **Linguagens Regulares (LR)** ou **Linguagens do Tipo 3**. São as linguagens mais simples.

**Resultado 3:**

Toda gramática do tipo 3 é também do tipo 2.

**Corolário 3:**

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

**Exemplo 1:**

$G_{11} = (\{S\}, \{a, b\}, P, S)$

$P = \{$

$S \rightarrow aS$

$S \rightarrow b \}$



Exemplo 1:

$G11 = (\{S\}, \{a, b\}, P, S)$

$P = \{$

$S \rightarrow aS$

$S \rightarrow b \}$

Resp.:  $L(G11) = \{a^n b \mid n \geq 0\}$  ou  $a^*b$

Exemplo 2:

$G_{12} = (\{S, A\}, \{a, b, c\}, P, S)$

$P = \{$

$S \rightarrow aS \mid bA$

$A \rightarrow c \}$

Exemplo 2:

$G_{12} = (\{S, A\}, \{a, b, c\}, P, S)$

$P = \{$

$S \rightarrow aS \mid bA$

$A \rightarrow c \}$

Resp.:  $L(G_{12}) = \{a^nbc \mid n \geq 0\}$  ou  $a^*bc$

## Formato BNF - Backus-Naur Form

### Exemplo 3:

$G_{13} = ( \{ \langle \text{Dig} \rangle, \langle \text{Int} \rangle \}, \{ +, -, 0, \dots, 9 \}, P, \langle \text{Int} \rangle )$   
 $P = \{$   
     $\langle \text{Int} \rangle ::= + \langle \text{Dig} \rangle \mid - \langle \text{Dig} \rangle$   
     $\langle \text{Dig} \rangle ::= 0 \langle \text{Dig} \rangle \mid 1 \langle \text{Dig} \rangle \mid \dots \mid 9 \langle \text{Dig} \rangle \mid 0 \mid 1 \mid 2 \mid \dots \mid 9$   
}

## Formato BNF - Backus-Naur Form

### Exemplo 3:

$$G13 = ( \{ \langle \text{Dig} \rangle, \langle \text{Int} \rangle \}, \{ +, -, 0, \dots, 9 \}, P, \langle \text{Int} \rangle )$$
$$P = \{ \langle \text{Int} \rangle ::= + \langle \text{Dig} \rangle \mid - \langle \text{Dig} \rangle$$
$$\langle \text{Dig} \rangle ::= 0 \langle \text{Dig} \rangle \mid 1 \langle \text{Dig} \rangle \mid \dots \mid 9 \langle \text{Dig} \rangle \mid 0 \mid 1 \mid$$
$$2 \mid \dots \mid 9 \}$$

Resp.:

$L(G13) = \text{conj. números inteiros com sinal}$   
 $\pm(0+1+2\dots+9)^+ \text{ ou } \pm(0..9)^+$

Modifique para que o sinal do número seja opcional

Exemplo 3:

$$G_{13} = ( \{ \langle \text{Dig} \rangle, \langle \text{Int} \rangle \}, \{ +, -, 0, \dots, 9 \}, P, \langle \text{Int} \rangle )$$
$$P = \{ \langle \text{Int} \rangle ::= + \langle \text{Dig} \rangle \mid - \langle \text{Dig} \rangle \mid \langle \text{Dig} \rangle$$
$$\langle \text{Dig} \rangle ::= 0 \langle \text{Dig} \rangle \mid 1 \langle \text{Dig} \rangle \mid \dots \mid 9 \langle \text{Dig} \rangle \mid 0 \mid 1 \mid$$
$$2 \mid \dots \mid 9 \}$$

Repare que, dessa forma, ela deixa de ser Regular!!!

Regular equivalente:

$$\langle \text{Int} \rangle ::= + \langle \text{Dig} \rangle \mid - \langle \text{Dig} \rangle \mid 0 \langle \text{Dig} \rangle \mid \dots \mid 9 \langle \text{Dig} \rangle \mid 0 \mid 1 \mid 2 \mid \dots \mid 9$$
$$\langle \text{Dig} \rangle ::= 0 \langle \text{Dig} \rangle \mid 1 \langle \text{Dig} \rangle \mid \dots \mid 9 \langle \text{Dig} \rangle \mid 1 \mid 2 \mid \dots \mid 9$$

Modifique para que a cadeia nula faça parte da linguagem

Exemplo 3:

$$G13 = ( \{ \langle \text{Dig} \rangle, \langle \text{Int} \rangle \}, \{ +, -, 0, \dots, 9 \}, P, \langle \text{Int} \rangle )$$
$$P = \{ \langle \text{Int} \rangle ::= +\langle \text{Dig} \rangle \mid -\langle \text{Dig} \rangle \mid \lambda$$
$$\langle \text{Dig} \rangle ::= 0\langle \text{Dig} \rangle \mid 1\langle \text{Dig} \rangle \mid \dots \mid 9\langle \text{Dig} \rangle \mid 0 \mid 1 \mid$$
$$2 \mid \dots \mid 9 \}$$

Modifique para que a cadeia nula faça parte da linguagem

Exemplo 3: **OU**

$$G13 = ( \{ \langle \text{Dig} \rangle, \langle \text{Int} \rangle \}, \{ +, -, 0, \dots, 9 \}, P, \langle \text{Int} \rangle )$$
$$P = \{ \langle \text{Int} \rangle ::= + \langle \text{Dig} \rangle \mid - \langle \text{Dig} \rangle$$
$$\langle \text{Dig} \rangle ::= 0 \langle \text{Dig} \rangle \mid 1 \langle \text{Dig} \rangle \mid \dots \mid 9 \langle \text{Dig} \rangle \mid \lambda \}$$



## Exemplo 4:

$G_{14} = ( \{A,B,C\}, \{0,1\}, P, A )$

$P = \{ A \rightarrow 0B \mid 0$

$B \rightarrow 1C$

$C \rightarrow 0B \mid 0 \}$

## Exemplo 4:

$G_{14} = ( \{A, B, C\}, \{0, 1\}, P, A )$

$P = \{ A \rightarrow 0B \mid 0$

$B \rightarrow 1C$

$C \rightarrow 0B \mid 0 \}$

**Resp.:**

**$L(G_{14}) = \{0(10)^*\}$**

# Linguagens Regulares e Expressões Regulares

- Expressões Regulares (ER) denotam conjuntos.
- Os conjuntos representados por ER coincidem com as Linguagens Regulares (LR).
- Assim, toda LR pode ser representada por uma ER.

# Expressões Regulares (ER)

Uma ER sobre um alfabeto  $\Sigma$  é definida como:

- a)  $\emptyset$  é uma ER e denota a linguagem vazia
- b)  $\lambda$  é uma ER e denota a linguagem contendo a palavra vazia, ie  $\{\lambda\}$
- c) Qualquer símbolo  $x \in \Sigma$  é uma ER e denota a linguagem  $\{x\}$
- d) Se  $r$  e  $s$  são ER denotando as linguagens  $R$  e  $S$  então:
  - $(r+s)$  ou  $(r|s)$  é ER e denota a linguagem  $R \cup S$
  - $(rs)$  é ER e denota a linguagem  $RS = \{w=uv \mid u \in R \text{ e } v \in S\}$
  - $(r^*)$  é ER e denota a linguagem  $R^*$

# Exemplos

- $00$  é uma ER denotando a linguagem  $\{00\}$
- $(0+1)^*$  denota a linguagem formada por todas as cadeias de 0's e 1's
- $(0+1)^* 00 (0+1)^*$  denota todas as cadeias de 0's e 1's com ao menos dois 0's consecutivos
- $a+b^*c$  denota um único  $a$  ou zero ou mais vezes  $b$  seguido de  $c$

- $(0+1)^* 001$  denota todas as cadeias de 0's e 1's terminadas em 001
- $0^*1^*2^*$  denota qualquer número de 0's seguido por qualquer número de 1's seguido por qualquer número de 2's
- $01^* + 10^*$  denota a linguagem consistindo de todas as cadeias que são um único 0 seguido por qualquer número de 1's OU um único 1 seguido por qualquer número de 0's.

## Omissão de parênteses

- Para omitir parênteses devemos respeitar:
  - O fecho (\*) tem prioridade sobre a concatenação (rs), que tem prioridade sobre a união.
  - A concatenação e a união são associadas da esquerda para a direita.
  - Ex:  $01^* + 1$  é agrupado como  $(0(1^*)) + 1 \Rightarrow L = \{1, 0, 01, 011, \dots\}$
- Usamos parênteses quando queremos alterar a prioridade:
- $(01)^* + 1 \Rightarrow L = \{1 \cup (01)^n \mid n \geq 0\} = \{1, \lambda, 01, 0101, \dots\}$
- $0(1^* + 1) \Rightarrow L = \{w \in \{0,1\}^* \mid w \text{ começa com } 0 \text{ seguido de } 1^n \mid n \geq 0\} \rightarrow \text{Lei distributiva à esq} = 01^* + 01 = \{0, 01, 011, 0111, \dots\}$

## Escreva a ER equivalente a:

- O conjunto de cadeias sobre  $\{0,1\}$  que termine com três 1's consecutivos.
- O conjunto de cadeias sobre  $\{0,1\}$  que tenha ao menos um 1.
- O conjunto de cadeias sobre  $\{0,1\}$  que tenha no máximo um 1.



## Escreva a ER equivalente a:

- O conjunto de cadeias sobre  $\{0,1\}$  que termine com três 1's consecutivos.

$$(0+1)^*111$$

- O conjunto de cadeias sobre  $\{0,1\}$  que tenha ao menos um 1.

$$(0+1)^*1(0+1)^*$$

- O conjunto de cadeias sobre  $\{0,1\}$  que tenha no máximo um 1.

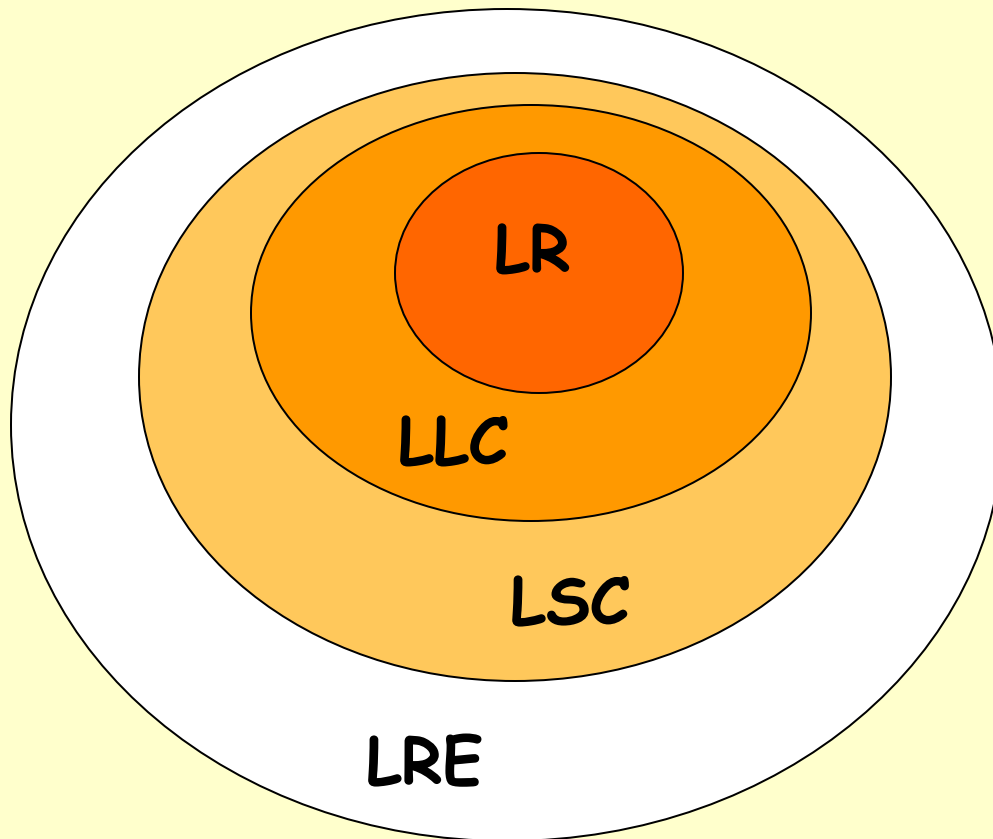
$$0^*(1+\lambda)0^*$$

### Hierarquia de Chomsky

Em termos gerais, para  $n \in \{0, 1, 2, 3\}$  pode-se afirmar que uma *linguagem* de qualquer tipo pode ser classificada também como sendo de tipo menor, de acordo com a **Hierarquia de Chomsky**.

Uma linguagem do tipo  $n$  é caracterizada pela existência de alguma gramática do tipo  $n$  que a descreva.

## Hierarquia de Chomsky



**LR = Linguagens  
Regulares**

**LLC = Linguagens  
Livres de Contexto**

**LSL = Linguagens  
Sensíveis ao Contexto**

**LEF = Linguagens  
Recursivamente  
Enumeráveis**

# Gramáticas e reconhecedores

Linguagens/Gramáticas	Reconhecedores
Rec. Enumerável	Máquina de Turing
Sensível ao contexto	Máquina de Turing com memória limitada
Livre de contexto	Autômato a pilha
Regular	Autômato finito

A partir de agora, vamos estudar as Linguagens (Problemas) mais simples: LR e LLC e seus reconhecedores.

## Linguagens e Reconhecedores

Linguagem	Gramática	Reconhecedor	Tempo para reconhecer $w$ ; $ w =n$
Tipo 0: Linguagens Computáveis ou Recursivamente Enumeráveis	Gramáticas com Estrutura de Frase	Máquinas de Turing	NP-completo
Tipo 1: Sensíveis ao Contexto	Gramáticas Sensíveis ao Contexto	Máquinas de Turing com memória limitada	Exponencial: $O(2^n)$
Tipo 2: Livres de Contexto	Gramáticas Livres de Contexto	Autômatos à Pilha	Polinomial Espaço: $O(n)$ ; Tempo: Geral: $O(n^3)$ ; Não-ambíguas: $O(n^2)$ ; Se $P = A \rightarrow aB$ ou $A \rightarrow Ba$ ou $A \rightarrow a$ : $O(n)$
Tipo 3: Conjuntos Regulares	Gramáticas Regulares	Autômatos Finitos	Linear: $O(n)$ e $O( E )$ (no tamanho do AF)

Classifique as gramáticas, dê a quádrupla e a  $L(G)$  e diga se são finitas/infinitas

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

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

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

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

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

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

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

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

9) Utilize o software JFLAP com os exemplos acima