LFA – Aula 09

Gramáticas e Linguagens Livres de Contexto

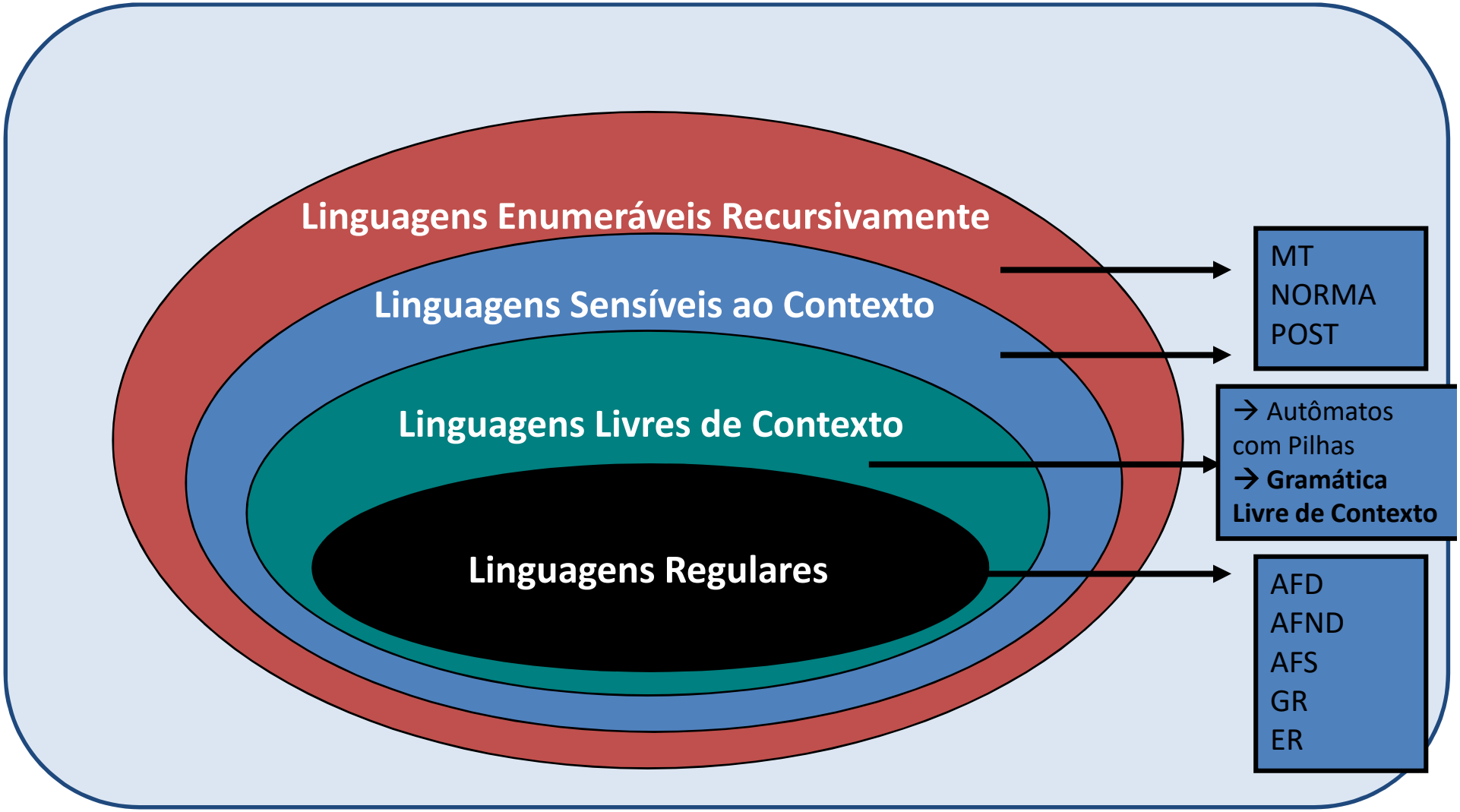
(Hopcroft, 2002)

Celso Olivete Júnior

olivete@fct.unesp.br

www.fct.unesp.br/docentes/dmec/olivete/lfa

Classes Gramaticais



Classes Gramaticais

a. Gramáticas com Estrutura de Frase ou Tipo 0: atuam no reconhecimento das Linguagens Enumeráveis Recursivamente

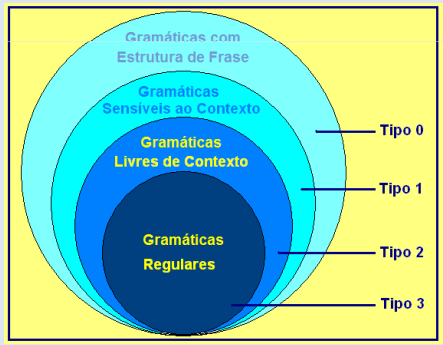
- São aquelas às quais nenhuma restrição é imposta.
  - Exemplo de reconhecedor: Máquina de Turing com fita de entrada infinita

- Produções da forma

  $\alpha \rightarrow \beta$

  Onde: $\alpha \in (Vn \cup Vt)^+$

  $\beta \in (Vn \cup Vt)^*$



G = (Vn, Vt, P, S)

  - Lado esquerdo da regra de produção pode conter N símbolos (terminais ou não terminais);
  - Lado direito da regra de produção pode conter N símbolos (terminais ou não terminais ou **vazio**);

Classes Gramaticais

a. Gramáticas com Estrutura de Frase ou Tipo 0

- Exemplo de GEF:

  G = ({A, B, C}, {a, b}, P, A)

  P: A → BC

  BC → CB

  B → b

  C → a

- Qual a linguagem gerada?
  - L(G) = {ba, ab}

Classes Gramaticais

b. Gramáticas Sensíveis ao Contexto ou Tipo 1: atuam no reconhecimento das Linguagens Sensíveis ao Contexto

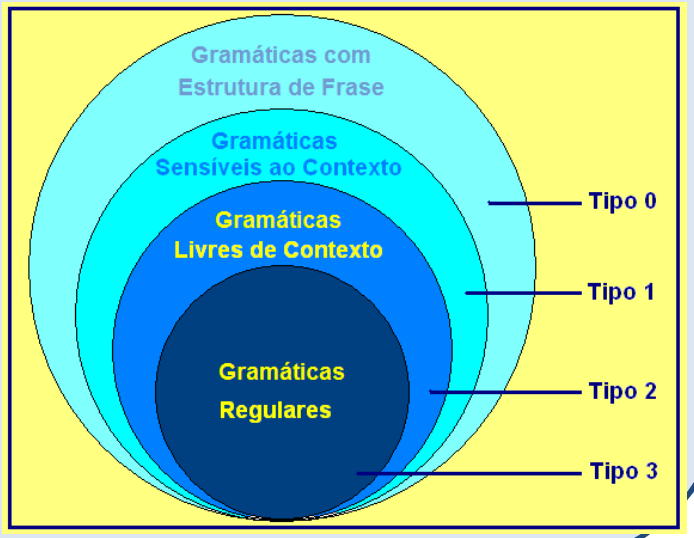
- Restrição: nenhuma substituição pode reduzir o comprimento da forma sentencial à qual a substituição é aplicada.
- Produções da forma

  $\alpha \rightarrow \beta$

  Onde: $\alpha \in (Vn \cup Vt)^+$

  $\beta \in (Vn \cup Vt)^+$

  $|\alpha| \leq |\beta|$



Classes Gramaticais

b. Gramáticas Sensíveis ao Contexto ou Tipo 1

- Exemplo de GSC:

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

P :
- $S \rightarrow aSBC \mid aBC$
- $CB \rightarrow BC$
- $aB \rightarrow ab$
- $bB \rightarrow bb$
- $bC \rightarrow bc$
- $cC \rightarrow cc$

$\alpha \rightarrow \beta$
Onde: $\alpha \in (Vn \cup Vt)^+$
$\beta \in (Vn \cup Vt)^+$
$|\alpha| \leq |\beta|$

Faça a derivação (mais à esquerda ou mais à direita)

- Qual a linguagem gerada?
  - $L(G) = \{a^n b^n c^n\}$

Classes Gramaticais

c. Gramáticas Livres de Contexto ou Tipo 2: atuam no reconhecimento das Linguagens Livres de Contexto

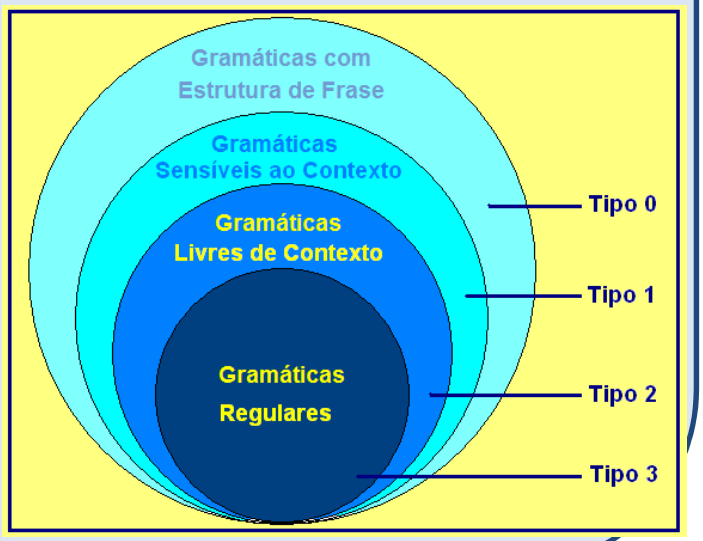
- As Gramáticas Livres de Contexto (GLC) ou do Tipo 2 são aquelas que no lado esquerdo da regra há apenas um símbolo não-terminal e no lado direito pode haver *n* terminais ou não-terminais.
- Produções da forma

  $\alpha \rightarrow \beta$

  Onde: $\alpha \in$ **(Vn)**

  $\beta \in$ **(Vn** $\cup$ **Vt)**$^+$

  |α| = 1 |β| > 0



Classes Gramaticais

c. Gramáticas Livres de Contexto ou Tipo 2

- Qual a linguagem gerada para:

G = ({S,A,B},{a,b},P,S)

P: S → AB

A → aA | a

B → bB | b

Faça a derivação (mais à esquerda ou mais à direita) para a entrada ***abbb***

$L(G)=\{a^n b^m\}$

Classes Gramaticais

d. Gramáticas Regulares ou Tipo 3: atuam no reconhecimento das Linguagens Regulares

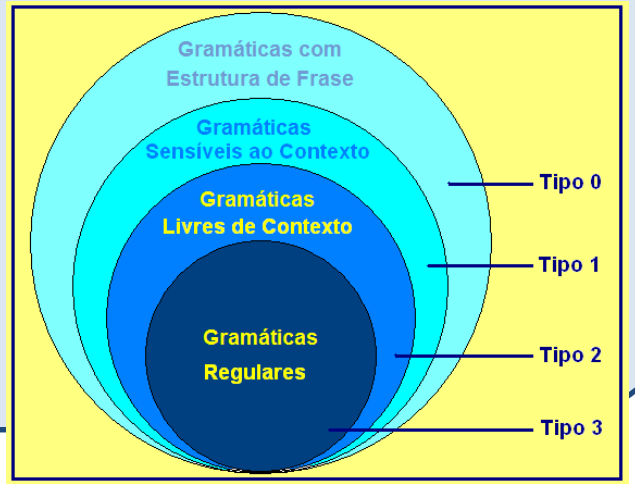
- Aplicando-se mais uma restrição sobre a forma das produções, pode-se criar uma nova classe de gramáticas, as Gramáticas Regulares (GR), de grande importância no estudo dos compiladores por possuírem propriedades adequadas para a obtenção de reconhecedores simples. Nas GRs, as produções são restritas às formas seguintes:

  $A \rightarrow aB$

  $A \rightarrow a$

  $A \rightarrow \varepsilon$

  onde $A,B \in Vn$ e $a \in Vt$, com $|A|=1$ e $|B| \leq 1$



Classes Gramaticais

d. Gramáticas Regulares ou Tipo 3

- Exemplo gramática em EBNF:

G = ({<Dig>, <Int>}, {+, -, 0, ..., 9}, P, <Int>)

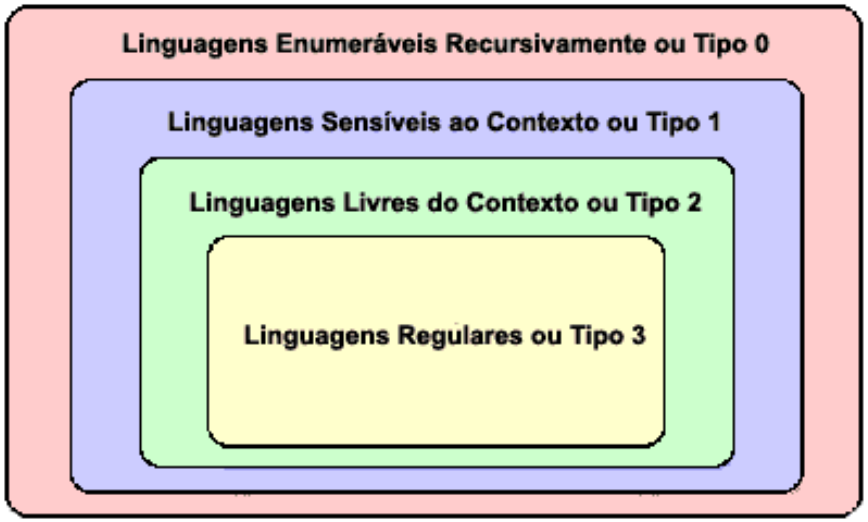
P: <Int> ::= +<Dig> | -<Dig>

<Dig> ::= 0<Dig> | 1<Dig>|...| 9<Dig> | 0 | 1 | 2 |...|9

- Qual linguagem gerada?
  - L(G) = conj. números inteiros com sinal ±[0..9]

Introdução

- Linguagens livre de contexto: abrange uma classe maior de linguagens.
  - A maior aplicação das gramáticas livres de contexto (GLC) ocorre na formalização sintática das linguagens de programação de alto nível;



Gramáticas livre de contexto

- Exemplo de utilização:
  - no processo de compilação → análise sintática
  - para descrever formatos de documentos (DTD), utilizados para troca de informações na Web (XML)

Gramáticas livre de contexto – aplicação na análise sintática

- Cada **linguagem** de programação **possui regras** que **descrevem** a **estrutura** sintática dos **programas**. **Em C**, por exemplo, um **programa é constituído por blocos**, um **bloco por comandos**, **comandos por expressões**, uma **expressão por tokens**, e assim por diante. Desta forma o **analisador sintático cuida** exclusivamente **da forma das sentenças da linguagem**, **baseando-se na gramática** que define a linguagem.

Gramáticas livre de contexto – exemplo

Exemplo 1: expressões constituídas por dígitos e sinais de mais e menos, como "9-5+2" e "3-1+6" podem ser descritas através da gramática:

<expr> → <num> | <num> + <expr> | <num> - <expr>
<num> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

ou

<expr> → <num>
<expr> → <num> + <expr>
<expr> → <num> - <expr>
<num> → 0
<num> → 1
<num> → ...
<num> → 9

•Gramática linguagem Pascal

<programa> --> program <identificador>; <comando> .
<comando> --> begin <comando> <resto_comando> |
<identificador> := <expressao_aritmetica> |
if <expressão> then <comando> else <comando> |
for <identificador> := <expressao_aritmetica> to <expressao> do <comando> |
repeat <comando> until <expressao> |
read <identificador> |
write <expressão_aritmetica> |
while <expressao> do <comando>
<resto_comando> --> ; <comando> <resto_comando> | ; end | end
<expressao> --> <expressao_aritmetica> | <expressao_logica>
<expressao_aritmetica> --> <termo> | <termo> <op> <expressão_aritmetica>
<expressao_logica> --> <termo> <comparacao> <expressão_logica>
<termo> --> <numero> | <identificador>
<op> --> + | - | / | *
<comparacao> --> < | <= | <> | > | >= | =
<numero> --> <digito> <numero> | <digito>
<digito> --> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<letra> --> a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<identificador> --> <letra> <numeroletra>
<numeroletra> --> <digito> <numeroletra> | <letra> <numeroletra> | ε

- Uma **gramática** livre de contexto é uma **notação formal** para **expressar uma linguagem**.

- Uma gramática consiste em uma ou mais variáveis que representam linguagens.

Gramáticas e Linguagens Livre de Contexto

• Formalmente as gramáticas são caracterizadas como quádruplas ordenadas

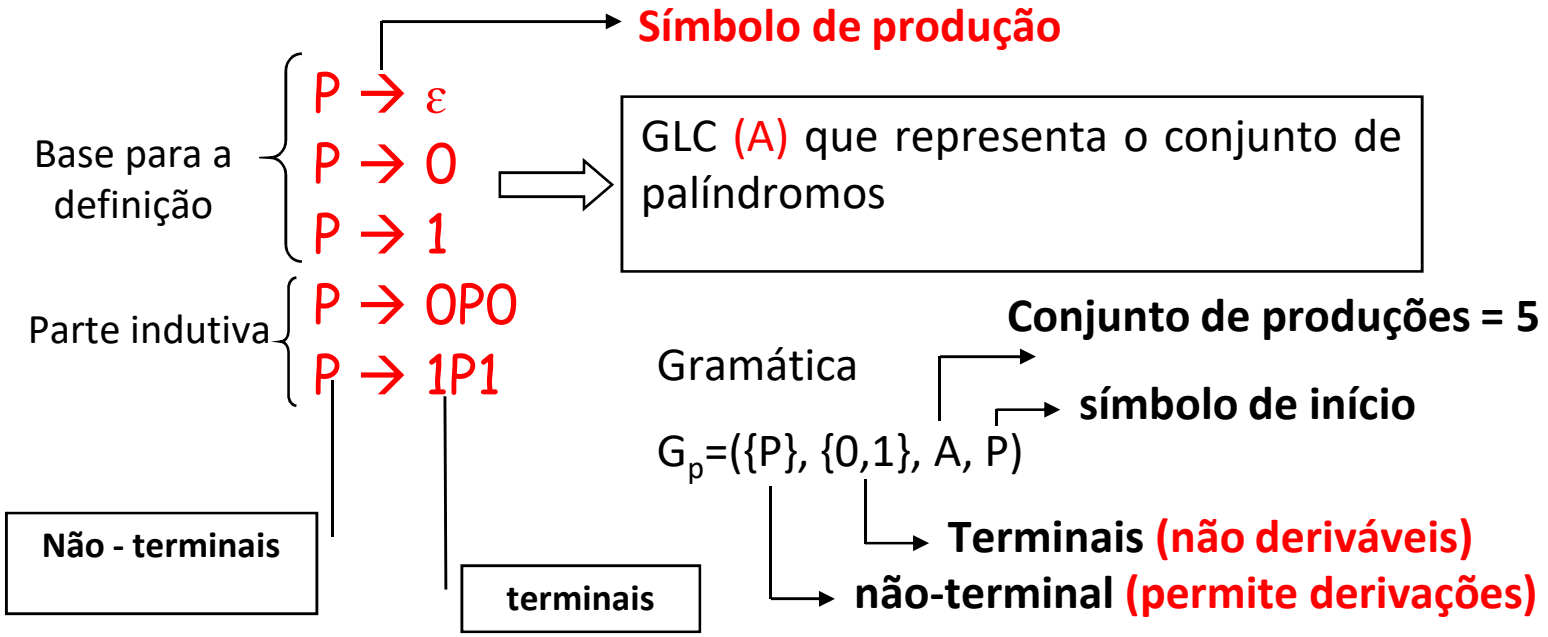
$$G = (\{V\}, T, P, S)$$

•onde:

- V representa o vocabulário não terminal da gramática - variáveis.
- T é o vocabulário terminal, contendo os símbolos que constituem as sentenças da linguagem.
- P representa o conjunto de todas as leis de formação (regras de produção) utilizadas pela gramática para definir a linguagem.
- S representa o símbolo de início

Gramáticas e Linguagens Livre de Contexto

- Ex: definição de uma GLC para ling. palíndromos
  - Caracteres base para o palíndromo: ε, 0 e 1
  - Se w é um palíndromo, então 0P0 e 1P1 também são.
  - Definição de uma GLC que representa o conjunto de palíndromos a partir dos caracteres base:



Gramáticas e Linguagens Livre de Contexto

- Exemplo 2: uma GLC que representa uma simplificação de expressões em uma linguagem de programação.
  - Operadores da linguagem + e *
  - Identificadores → formados por letras seguidas de zero ou mais letras e dígitos

    (a + b) (a + b + 0 + 1)*
  - Letras são formadas apenas por a e b
  - Dígitos apenas por 0 e 1
  - Variáveis da gramática: E (expressões) e I (identificadores)
  - Conjunto de Produções que representam essa ER

1. E → I	-> base
2. E → E + E	-> E conectada por um sinal de adição
3. E → E * E	-> E conectado com o simb. multiplicação
4. E → (E)	-> E entre parênteses
5. I → a	-> identificador

6. I → b	->identificador
7. I → Ia	
8. I → Ib	
9. I → I0	
10. I → I1	

Gramáticas e Linguagens Livre de Contexto

- Notação / Convenções
  - Variáveis: letras do alfabeto maiúsculas {A,B,…,Z}
  - Terminais: letras do início do alfabeto minúsculas {a,b,c…}, dígitos {0..9} e outros caracteres como +, -, *, /
  - Não-Terminais: letras do fim do alfabeto maiúsculas, como X ou Y, são terminais ou variáveis

Gramáticas e Linguagens Livre de Contexto

- Exercícios

1. Crie as regras de produção (P) e defina a gramática G (quádrupla) das linguagens abaixo:
   a) Uma linguagem que define expressões envolvendo elementos de 0 a 9, somas, subtrações, multiplicações, divisões e expressões entre parênteses.
   b) L(G) = {ba,ab}
   c) L(G) = $\{0^n1^m \mid n,m >= 0\}$ ou 0*1*
   d) L(G) = $\{(01)^n \mid n >= 1\}$
   e) L(G) = $\{(011)^n \mid n >= 1\}$
   f) L(G) = $\{a^nb^nc^i \mid n >= 1 \text{ e } i >= 0\}$

As Gramáticas Livres de Contexto (GLC) ou do Tipo 2 são aquelas que no lado esquerdo da regra há apenas um símbolo não-terminal. Do lado direito pode existir ***n*** não-terminais e terminais (***n>=0***)

Gramáticas e Linguagens Livre de Contexto

- Resolução

a) Uma linguagem que define expressões envolvendo elementos de 0 a 9, somas, subtrações, multiplicação, divisões e expressões entre parênteses.

G=({E,D}, {0..9,+, *, -, /, (,)}, P, E)

1. E → E + E
2. E → E - E
3. E → E * E
4. E → E / E
5. E → (E)
6. E → D
7. D → 0
8. D → 1
9. D → ...9

GLC: reconhecimento de cadeias

- Exemplo:

1. S → A1B
2. A → 0A | ∈
3. B → 0B | 1B | ∈

  - O processo de reconhecimento de uma cadeia por uma GLC pode ser feita fazendo a leitura das regras de produção, desde a primeira até a última regra. Esse processo é conhecido por *inferência recursiva*.
  - No exemplo acima começamos com uma regra que nos leva a um A (não-terminal) e esse A é formado por 0 seguido de um A, que pode ficar em loop (A*) ou não aparecer nenhuma vez (∈), seguido de 1 resultando em **0*1**
  - Seguido de um B, que nos leva a um 0B ou 1B. Esse B é um não terminal que pode aparecer uma, nenhuma ou várias vezes, resultando em **(0 + 1)***
  - Linguagem aceita pela GLC = **0*1 (0 + 1)***

GLC: reconhecimento de cadeias

- Existe uma outra maneira de fazer o reconhecimento, conhecida por *derivação*, onde é feita uma expansão (derivação) de uma das primeiras regras de produção que formam a base. Esse processo é definido pelo símbolo ⇒

1. E → I	6. I → b
2. E → E + E	7. I → Ia
3. E → E * E	8. I → Ib
4. E → (E)	9. I → I0
5. I → a	10. I → I1

Exemplo: reconhecer a ER = **a * (a + b00)**

Ex:

E ⇒ E * E ⇒ I * E ⇒ a * E ⇒
a * (E) ⇒ a * (E + E) ⇒ a * (I + E) ⇒
a * (a + E) ⇒ a * (a + I) ⇒

a * (a + I0) ⇒ a * (a + I00) ⇒

a * (a + b00)

• ***Observe*** ***que no decorrer da substituição prevaleceu a substituição de uma variável mais à esquerda***

GLC: reconhecimento de cadeias

- O processo de *derivação* pode ocorrer mais à esquerda ou mais à direita.
  - Dá-se o nome de GLC com **derivação à esquerda** e GLC com **derivação à direita**

Derivação mais à esquerda

- A variável mais à esquerda de uma string sempre é substituída por uma regra de produção. Usa-se o símbolo $\underset{E}{\Rightarrow}$ para representar essa derivação. Ex: derivar mais à esquerda a produção para gerar a ER = a * (a + b00)

1. E → I
2. E → E + E
3. E → E * E
4. E → (E)
5. I → a

6. I → b
7. I → Ia
8. I → Ib
9. I → I0
10. I → I1

Ex:

$E \underset{E}{\Rightarrow} E * E \underset{E}{\Rightarrow} I * E \underset{E}{\Rightarrow} a * E \underset{E}{\Rightarrow}$

$a * (E) \underset{E}{\Rightarrow} a * (E + E) \underset{E}{\Rightarrow} a * (I + E) \underset{E}{\Rightarrow}$

$a * (a + E) \underset{E}{\Rightarrow} a * (a + I) \underset{E}{\Rightarrow}$

$a * (a + I0) \underset{E}{\Rightarrow} a * (a + I00) \underset{E}{\Rightarrow}$

$a * (a + b00)$

Derivação mais à direita

- A variável mais à direita de uma string sempre é substituída por uma regra de produção. Usa-se $\underset{D}{\Rightarrow}$ o símbolo para representar essa derivação. Ex: ER = a * (a + b00)

1. E → I
2. E → E + E
3. E → E * E
4. E → (E)
5. I → a

6. I → b
7. I → Ia
8. I → Ib
9. I → I0
10. I → I1

Ex:

$E \underset{D}{\Rightarrow} E * E \underset{D}{\Rightarrow} E * (E) \underset{D}{\Rightarrow} E * (E + E) \underset{D}{\Rightarrow}$

$E * (E + I) \underset{D}{\Rightarrow} E * (E + I0) \underset{D}{\Rightarrow}$

$E * (E + I00) \underset{D}{\Rightarrow} E * (E + b00) \underset{D}{\Rightarrow}$

$E * (I + b00) \underset{D}{\Rightarrow} E * (a + b00) \underset{D}{\Rightarrow}$ I

$* (a + b00) \underset{D}{\Rightarrow} a * (a + b00)$

Formas sentenciais de uma GLC

- As derivações à partir do símbolo início da produção são conhecidas como formas sentenciais. Podendo ser uma *forma sentencial à esquerda* ou *uma forma sentencial à direita*.
- Exemplo de forma sentencial à esquerda

Ex:

$$E \underset{E}{\Rightarrow} E * E \underset{E}{\Rightarrow} I * E \underset{E}{\Rightarrow} a * E$$

- Forma sentencial à direita

Ex:

$$E \underset{D}{\Rightarrow} E * E \underset{D}{\Rightarrow} E * (E) \underset{D}{\Rightarrow} E * (E + E)$$

Exercícios

2) Exercício 5.1.2 e 5.1.4 (pág. 192)

3) Forneça a linguagem gerada e a Gramática da seguinte regra de produção

1. A → 0B | 0
2. B → 1C
3. C → 0B | 0

4) A partir da Gramática (G) e da produção (P) abaixo, aplique o processo de derivação (à esquerda ou à direita) para saber qual é a linguagem gerada.

G = ({B}, {0,1},P,S)

P

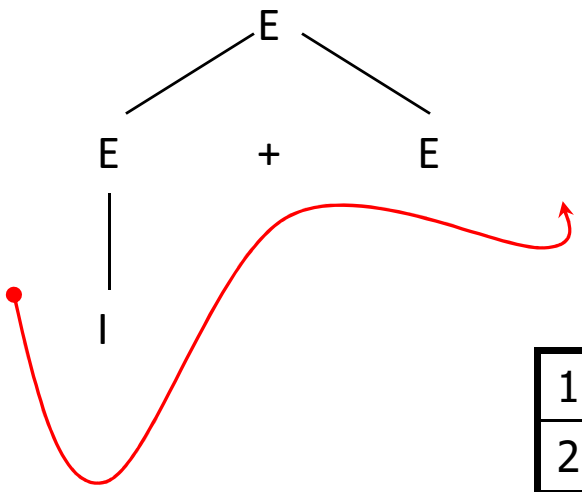
1. S → 0B1
2. B → 01

Árvores de análise sintática

- As derivações de uma gramática também podem ser representadas através de árvores.
- Construção da árvore de análise sintática
  1. Cada nó é rotulado por uma variável em V
  2. Cada folha é rotulada por uma variável, um terminal ou ε, ela deve ser o único filho do seu pai
  3. Se um nó é rotulado por A e seus filhos são rotulados por $X_1, X_2, X_3, \ldots, X_k$. Então: $A \rightarrow X_1, X_2, X_3, \ldots, X_k$ é uma produção em P. O único momento em que ε pode aparecer é quando $A \rightarrow \varepsilon$ for uma produção em G

Árvores de Análise Sintática

- Exemplo 1: árvore mostrando a derivação de I + E a partir de E



1. E → I
2. E → E + E
3. E → E * E
4. E → (E)
5. I → a

6. I → b
7. I → Ia
8. I → Ib
9. I → I0
10. I → I1

Árvores de Análise Sintática

- Exemplo 2: árvore mostrando a derivação para a gramática de palíndromos

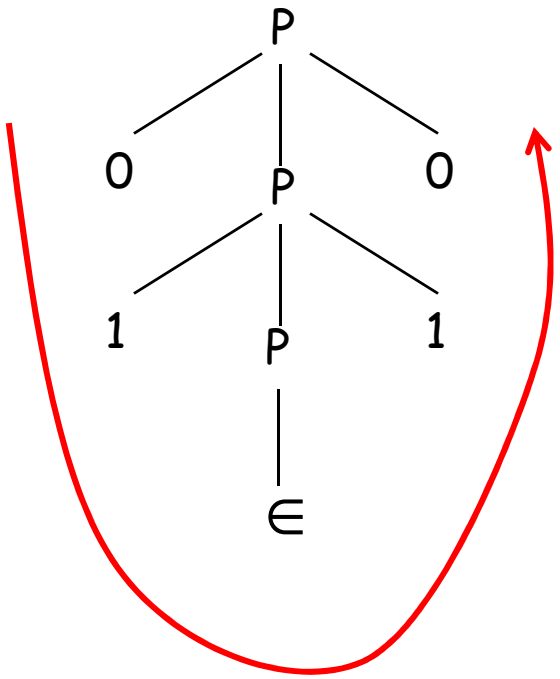
$P \rightarrow \varepsilon$

$P \rightarrow 0$

$P \rightarrow 1$

$P \rightarrow 0P0$

$P \rightarrow 1P1$



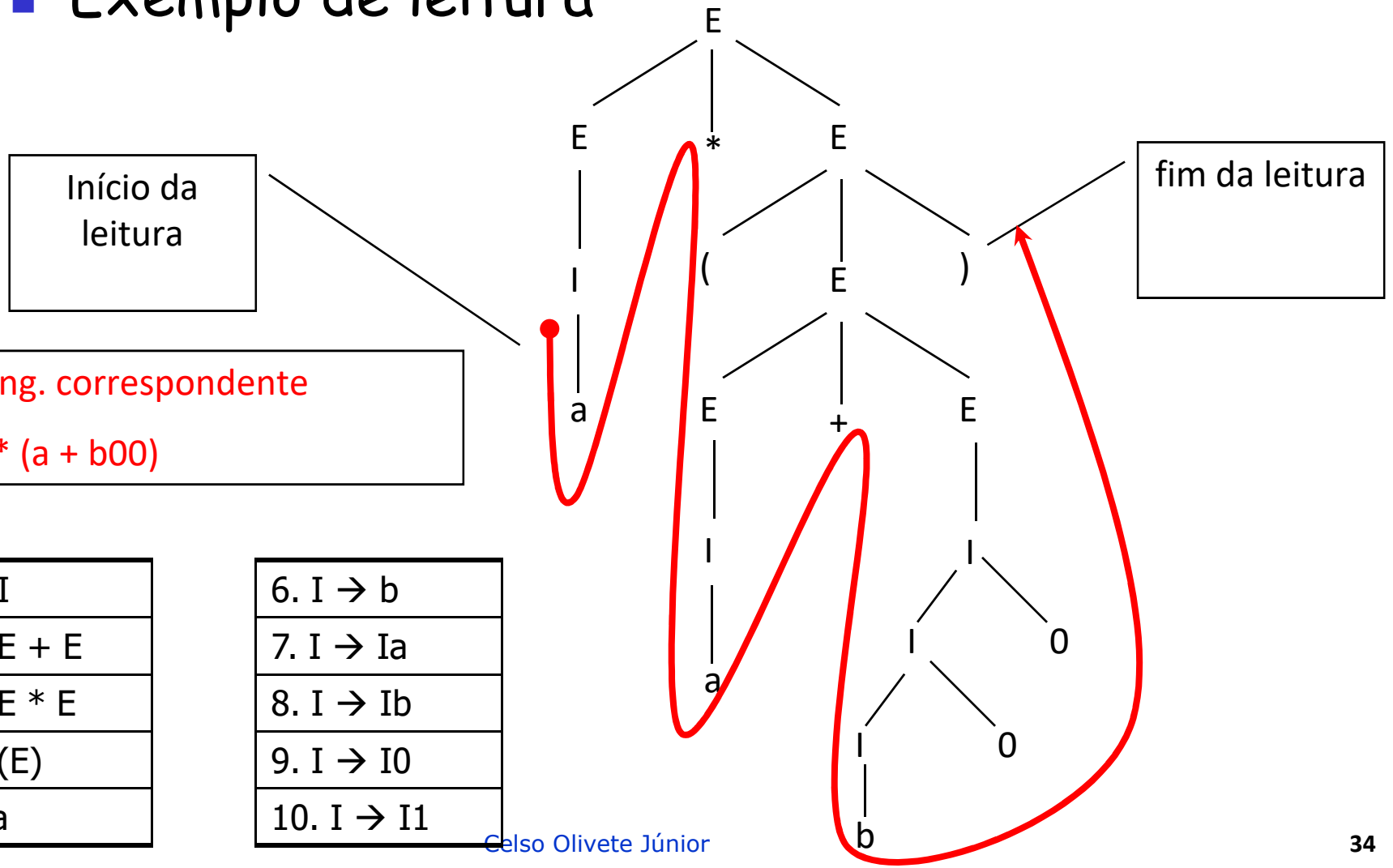
Entrada: 0110

O resultado de uma árvore de análise sintática

- A leitura de uma árvore é feita a partir da concatenação das folhas a partir da esquerda. O resultado dessa leitura é conhecido por *resultado da árvore* (que é um string derivado a partir da variável raiz)

O resultado de uma árvore de análise sintática

- Exemplo de leitura



1. E → I
2. E → E + E
3. E → E * E
4. E → (E)
5. I → a

6. I → b
7. I → Ia
8. I → Ib
9. I → I0
10. I → I1

Exercícios

5) Exercício – resolver o exercício 5.2.1 (pág. 205)

6) Construa a árvore de análise sintática para a gramática do exercício 3 e 4.

7) Considere a gramática G = ({S}, {a, b}, {S → ab; S → ba; S→SS; S→ aSb; S→bSa}, S).
   a) Determine, justificando, a linguagem gerada pela gramática G.
   b) Determine uma derivação à esquerda, caso exista, da seguinte sentença: **aababbba**

8) Considere a gramática G = ({ S }, { 0, 1 }, { S → 0S0 ; S → 1X ; X→1X; X → 1 }, S). Determine, justificando, a linguagem gerada pela gramática G.

Exercícios

9) Considere a gramática G = (S, T, P, A) que representa o cabeçalho de métodos na linguagem Java (sem os modificadores de acesso), onde

Σ = { S, Type, Param, Exception, ParamList }

T={ id, int, boolean, (,), ,, throws, ArithmeticException }

P é formado pelas seguintes produções :

S →Type id (Param) Exception

Type →int | boolean

Param → ε | ParamList

ParamList →Type id | ParamList , Type id

Exception → ε | throws ArithmeticException

a)Encontre a derivação mais à esquerda de

int id (boolean id) throws ArithmeticException

Exercícios

10) Considere a seguinte gramática G = (S, T, P, A) que descreve uma versão simplificada de escrita de documentos na linguagem *MathML (Mathematical Markup Language)*, onde

Σ = { D, A, C }

T = { <, >, /, ci, cn, apply, =, id, string, plus, sin, number }

P é formado pelas seguintes produções :

D → < ci A > C < / ci > | < cn > number < / cn > | < apply > C < / apply >

A → ε | A id = string

C → id | < sin / > D | < plus / > D D

a)Encontre a derivação mais à esquerda para

< apply > < plus / > < ci > id < / ci > < cn > number < / cn > < / apply >