

# Gramáticas Livres de Contexto

Marcelo Johann

## Conteúdo da aula

- Trabalho - primeira etapa
  - Definição, código e estrutura
  - Mais elementos e detalhes de lex
- GLCs
  - Gramática, produção, derivações, árvores
  - Ambíguas, sem ciclos,  $\epsilon$ -livres, fatoradas à esquerda, recursivas à esquerda, simplificadas
- Transformações
  - Eliminação de produções vazias, de recursividade à esquerda (direta, indireta), fatoração

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 2

## Hierarquia de Linguagens de Chomsky

|        |                            |   |
|--------|----------------------------|---|
| Tipo 0 | Sem restrição recur. enum. | Qualquer $\underline{u} \rightarrow \underline{v}$ desde que $\underline{u}$ seja não vazio                               |
| Tipo 1 | Sensível ao contexto       | $\underline{w}A\underline{z} \rightarrow \underline{w}V\underline{z}$<br>( $\underline{w}, \underline{z}$ são o contexto) |
| Tipo 2 | Livre do Contexto          | $A \rightarrow \underline{v}$<br>(1 símbolo a esquerda)   |
| Tipo 3 | regular                    | $A \rightarrow a \mid aB \mid \epsilon$<br>(derivações a direita)   |

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 3

## Definições

- Gramáticas
- Produção
- Derivações
- Árvores de Derivação

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 4

## Usando Gramáticas para definir a sintaxe

- Muitas construções de linguagens de programação são recursivas
  - Regexp não podem reconhecer tais construções.
- Exemplo de definição recursiva:
  - se  $S_1$  e  $S_2$  são enunciados e  $E$  é uma expressão, então "if  $E$  then  $S_1$  else  $S_2$ " é um enunciado
- Exemplo de definição com uma gramática:
  - cmd  $\rightarrow$  if expr then cmd else cmd
  - expr  $\rightarrow$  ( bool ) | expr && expr | expr || expr
  - bool  $\rightarrow$  true | false

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 5

## Gramáticas formais: Notações e Definições

- Uma Gramática Formal é um conjunto de 4 elementos:
  - $G := \{S, P, N, T\}$
- $S$ : o Símbolo Inicial ( $S \in N$ )
- $P$ : Conjunto de regras de produções do tipo  $\underline{u} \rightarrow \underline{v}$
- $T$ : conjunto de símbolos terminais
  - Palavras ou tokens da linguagem
- $N$ : conjunto de símbolos não terminais
  - Símbolos que podem ser substituídos pelas produções
- Vocabulário e notações:
  - $V$ : alfabeto
    - $V = N \cup T$  ( $N \cap T = \emptyset$ )
  - $\underline{u}$ : string pertencente a  $V$

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 6

## Derivações

- Seja:
  - um não-terminal  $A \in N$ ,
  - uma regra de produção  $p = \{A \rightarrow v\} \in P$ , e
  - $w, z$  dois strings quaisquer.
 A transformação:
 
$$wAz \Rightarrow wvz$$
 É chamada "**derivação** em um passo usando  $p$ "
- Se  $w_1 \Rightarrow w_2 \Rightarrow w_3 \dots \Rightarrow w_n$  então podemos dizer
 
$$w_1 \Rightarrow^* w_n$$
 – "derivação em múltiplos (0 ou mais) passos"; também  $\Rightarrow^*$  (um ou mais)

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 7

## Definição da Linguagem Gerada

- A **linguagem gerada  $G$**  (notada  $L(G)$ ) é:
  - O conjunto formado por **todos os strings** de símbolos **terminais deriváveis a partir** do símbolo inicial  $S$
  - $L(G) = \{s \mid s \text{ é um elemento de } T^* \text{ e } S \Rightarrow^+ s\}$
- Equivalência:**
  - $G_1$  e  $G_2$  são equivalentes se  $L(G_1) = L(G_2)$ 
    - $\Rightarrow$  Todos strings gerados estão em  $L$
    - $\Leftarrow$  Todos strings  $w \in L$  podem ser gerados
- Convenções**
  - Símbolos que representam terminais em minúsculos:
    - $a, v, x, y, \dots$
  - Símbolos que representam não-terminais em maiúsculos:
    - $X, Y, \text{TERM}, S, \dots$
  - Símbolos que representam formas sentenciais (seqüências de terminais e não-terminais): letras gregas ou sublinhadas:
    - $\alpha, \beta, \omega, \underline{w}, \underline{z}$

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 8

## Caso particular: Gramática Regular

- Gramática regular:**
  - produções exclusivamente da forma:
    - $A \rightarrow wB$
    - $A \rightarrow w$ ,
    - onde  $w \in T^*$  e  $A, B \in N$  (gramática linear à direita)
  - A linguagem gerada por uma gramática regular é regular.
  - Pode-se gerar os mesmos *strings* através de uma expressão regular.

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 9

## Exemplo $G = \{N, T, P, S\}$

$N = \{S, B, C, D\}$

$T = \{0, 1\}$

$S \rightarrow 0B \mid 1C$   
 $B \rightarrow 1S$   
 $C \rightarrow 0C \mid 0D \mid 1$   
 $D \rightarrow 0$

$S$

**Exercício:** fornecer expressão regular equivalente à gramática acima.

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 10

## Árvore de derivação (Parse Tree)

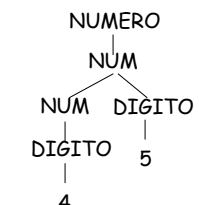
- Árvore de derivação:**
  - representação gráfica da derivação de uma sentença (*string*)
  - estrutura hierárquica que originou a sentença
    - A raiz da árvore representa o símbolo inicial
    - Os vértices interiores são não-terminais
    - Os símbolos terminais e a palavra vazia são folhas

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 11

## Árvore de derivação – exemplo 1

- Gramática  $G = ($ 
  - $\{ \text{NUMERO, NUM, DIGITO} \}$ ,
  - $\{0, 1, 2, \dots, 9\}$ ,
  - $P$ ,
  - $\text{NUMERO}$  )
- $P$  (regras de produção) =
  - $\text{NUMERO} \rightarrow \text{NUM}$
  - $\text{NUM} \rightarrow \text{NUM DIGITO} \mid \text{DIGITO}$
  - $\text{DIGITO} \rightarrow 0 \mid 1 \mid \dots \mid 9$



Árvore de derivação para a sentença 45

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 12

## Derivação mais à esquerda e mais à direita

- **Derivação mais à esquerda de uma sentença:**
  - sequência de formas sentenciais que se obtém derivando sempre o símbolo não-terminal mais à esquerda.
- **Derivação mais à direita de uma sentença:**
  - sequência de formas sentenciais que se obtém derivando sempre o símbolo não-terminal mais à direita.

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 13

## Árvore de derivação – exemplo 2

Gramática  $G = (\{E\}, \{+, -, *, /, (, ), x\}, P, E)$  sendo

$P = \{E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid x\}$

A mesma árvore pode representar mais de uma derivação para uma mesma sentença

Possíveis derivações para a árvore:

$E \Rightarrow E + E \Rightarrow x + E \Rightarrow x + E * E \Rightarrow x + x * E \Rightarrow x + x * x$   
 $E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow E + E * x \Rightarrow E + x * x \Rightarrow x + x * x$   
 $E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow x + E * E \Rightarrow x + x * E \Rightarrow x + x * x$

Isso não é problema....

MAS....

INF01033 - Compiladores B - Marcelo Johann - 2012/1

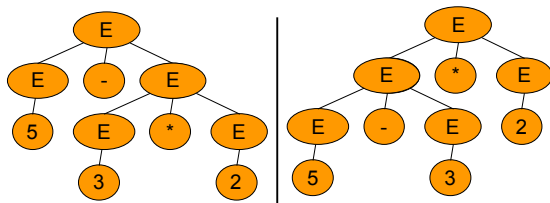
Aula 06 : Slide 14

## Gramática Ambígua

$E \rightarrow E \text{ OP } E \mid \text{"(" E ")} \mid x$   
 $\text{OP} \rightarrow \text{"*"} \mid \text{" / " } \mid \text{" + " } \mid \text{" - " }$

- Considere  $5 - 3 * 2$

Quando há mais de uma árvore de derivação para uma mesma sentença



INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 15

## Tipos ou Características

- Gramáticas Ambíguas
- Gramáticas sem ciclos
- Gramáticas  $\epsilon$ -livres
- Gramáticas fatoradas à esquerda
- Gramáticas recursivas à esquerda
- Gramáticas simplificadas

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 16

## Outras Classificações de Gramáticas

- **Gramática sem ciclos:**
  - Uma gramática sem ciclos é uma GLC que não possui derivações da forma  $A \Rightarrow^+ A$  para algum  $A \in N$
- **Gramática  $\epsilon$ -livre :**
  - GLC que não possui produções vazias do tipo  $A \rightarrow \epsilon$  **exceto** a produção  $S \rightarrow \epsilon$  (S é o símbolo inicial).

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 17

## Outras Classificações de Gramáticas

- **Gramática fatorada à esquerda:**
  - GLC que **não** possui produções do tipo  $A \rightarrow \alpha\beta_1\alpha\beta_2$  para alguma forma sentencial  $\alpha$ .
- **Gramática recursiva à esquerda:**
  - GLC que permite a derivação  $A \Rightarrow^+ A\alpha$  para algum  $A \in N$
  - O não terminal A deriva ele mesmo, de forma **direta** ou **indireta**, como

OBS: RECONHECEDOR TOP-DOWN não aceita gramáticas recursivas à esquerda

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 18

## Transformações de GLCs

- (A) Eliminação de produções vazias
- (B) Eliminação de recursividade à esquerda:
  - recursão direta
  - recursão indireta
- (C) Fatoração de uma gramática

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 19

## Eliminação de Produções Vazias

- **Objetivo:**
  - eliminar produções da forma  $A \rightarrow \epsilon$ .
- **Algoritmo:** seja  $G = (N, T, P, S)$  uma GLC
  - **Etapa 1:**
    - construir  $N_\epsilon$ , o conjunto de não-terminais que geram a palavra vazia:
 
$$N_\epsilon = \{A \mid A \rightarrow \epsilon\};$$
 // é um conjunto de símbolos
    - Repita
 
$$N_\epsilon = N_\epsilon \cup \{X \mid X \rightarrow X_1 \dots X_n \in P \text{ tq } X_1, \dots, X_n \in N_\epsilon\}$$
    - Até que o cardinal de  $N_\epsilon$  não aumente.

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 20

## Eliminação de Produções Vazias

- **Etapa 2:**
  - construir o conjunto de produções sem produções vazias:
 
$$P_1 = \{A \rightarrow \alpha \mid \alpha \neq \epsilon\};$$
  - Repetir
    - Para toda produção  $A \rightarrow \alpha \in P_1$  e  $X \in N_\epsilon$  tal que  $\alpha = \alpha_1 X \alpha_2$  e  $\alpha_1 \alpha_2 \neq \epsilon$
    - Faça  $P_1 = P_1 \cup \{A \rightarrow \alpha_1 \alpha_2\}$
    - Até que o cardinal de  $P_1$  não aumente

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 21

## Eliminação de Produções Vazias

- **Etapa 3:**
  - incluir a geração da palavra vazia, se necessário:
  - Se a palavra vazia pertence à linguagem, então a gramática resultante é
 
$$G_2 = (N, T, P_2, S), \text{ onde } P_2 = P_1 \cup \{S \rightarrow \epsilon\}$$

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 22

## (B) Eliminação de recursividade à esquerda

- Exemplo de GLC recursiva à esquerda:
 
$$A \rightarrow Aa \mid b$$
- Gramáticas transformadas equivalentes:

Com a palavra vazia  
 $A \rightarrow bX$   
 $X \rightarrow aX \mid \epsilon$

Sem a palavra vazia  
 $A \rightarrow b \mid bX$   
 $X \rightarrow a \mid aX$

Obs: pode ainda haver recursão indireta!

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 23

## (B) Outro exemplo de recursividade

- $E \rightarrow E+T \mid T$
- $T \rightarrow T^*F \mid F$
- $F \rightarrow (E) \mid Id$

A regra  $E \rightarrow E+T \mid T$  se torna:  
 $E \rightarrow TE'$   
 $E' \rightarrow +TE' \mid \epsilon$

A regra  $T \rightarrow T^*F \mid F$  se torna:  
 $T \rightarrow FT'$   
 $T' \rightarrow *FT' \mid \epsilon$

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 24

### (C) Fatoração de uma gramática

- Elimina a indecisão de qual produção aplicar quando duas ou mais produções iniciam com a mesma forma sentencial

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$$

Se torna:

$$A \rightarrow \alpha X$$

$$X \rightarrow \beta_1 \mid \beta_2$$

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 25

### (C) Exemplo de Fatoração a Esquerda

$\text{Cmd} \rightarrow \text{if Expr then Cmd else Cmd}$

$\text{Cmd} \rightarrow \text{if Expr then Cmd}$

$\text{Cmd} \rightarrow \text{Outro}$

- Fatorando a esquerda:

$\text{Cmd} \rightarrow \text{if Expr then Cmd ElseOpc}$

$\text{Cmd} \rightarrow \text{Outro}$

$\text{ElseOpc} \rightarrow \text{else Cmd} \mid \epsilon$

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 26

### Leituras e Tarefas

Fazer exercícios do livro.

INF01033 - Compiladores B - Marcelo Johann - 2012/1

Aula 06 : Slide 27