

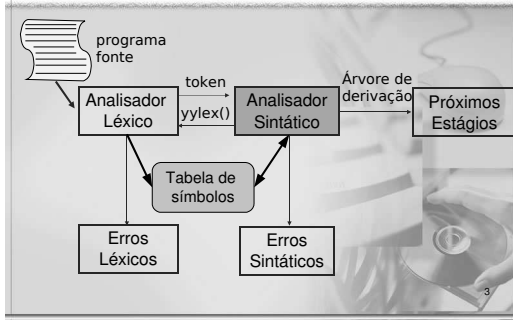
# Compiladores

Análise sintática (1)  
Revisão:  
Gramáticas Livres de Contexto

## Plano da aula

- Introdução: porque a análise sintática?
- Noções sobre Gramáticas Livres de Contexto:
  - Definição
  - Propriedades
  - Derivações
    - Árvore de derivação
  - Transformações de GLC
    - Eliminação de  $\epsilon$
    - Eliminação da recursão à esquerda
    - Fatoração

## Análise Sintática



## Os erros típicos são sintáticos

Ripley e Druseikis [78]

- Erros em uma amostra de programas Pascal
- 60% apresentados eram corretos (semântica e sintaticamente)
- Erros
  - 80% das sentenças erradas continham apenas um erro
  - 13% das sentenças erradas continham dois erro
  - 90% em erros de token
  - 60% em erro de pontuação (como uso incorreto do ;)
  - 20% em erros com operadores ou operandos
  - 15% palavras-chave
  - 5% outros erros

```
1. program pmax(input, output) ;
2. var
3.   x, y : integer;
4. function max (i: integer, j: integer) : integer
5.   { retorna o maior de i e j }
6. begin
7.   if i > j then max = i;
8.   else max := j
9. end;
10.
11. readln (x, y);
12. writeln (max (x, y))
13. end.
```

## Os erros típicos são sintáticos

- **Linha 4 :**
    - \* uso de , ao invés de ;
  - **Linha 7 :**
    - \* uso de ; antes de else
  - **Linha 8 :**
    - \* uso de = ao invés de :=
  - **Linha 13 :**
    - \* writeln ao invés de writeln
- ```
1. program pmax(input, output) X
2. var
3.   x, y : integer;
4. function max (i: integer, j: integer) : integer
5.   { retorna o maior de i e j }
6. begin
7.   if i > j then max := i;
8.   else max X= j
9. end;
10. X
11. readln (x, y);
12. writeln (max (x, y))
13. end.
```

## Usando Gramáticas

- Muitas construções de linguagens de programação são recursivas
- 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
- Usando gramáticas:
  - cmd  $\rightarrow$  if expr then cmd else cmd

## Notação e Definições

- Gramática Formal
  - $G := \{T, N, P, S\}$
- S: Símbolo Inicial ( $S \in N$ )
- P: Conjunto de produções
  - $u \rightarrow v$
- T: conjunto de terminais
  - Palavras ou tokens da linguagem
- N: conjunto de não terminais
  - Símbolos que podem ser substituídos
- Vocabulário:
  - V: alfabeto
    - $V = N \cup T$  ( $N \cap T = \emptyset$ )
    - $u$ : string pertencente a V
    - Símbolo Vazio  $\epsilon$
    - BNF: Backus-Naur form
      - $\langle u \rangle ::= \langle v \rangle \mid \langle w \rangle$
      - Igual a ( $u = v$ ,  $u = w$ ) em P

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

|                                   |                       |
|-----------------------------------|-----------------------|
| $T = \{0, 1\}$                    | Os terminais          |
| $N = \{S, B, C, D\}$              | Os não terminais      |
| $S \rightarrow 0B \mid 1C$        | As regras de produção |
| $B \rightarrow 1S$                |                       |
| $C \rightarrow 0C \mid 0D \mid 1$ |                       |
| $D \rightarrow 0$                 |                       |
| S                                 | O símbolo de entrada  |

Fornecer expressão regular para a gramática acima ?  
 $(01)^*10^*(00 \mid 1)$

## Definição de 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)

## Definição de Linguagem Gerada

- A linguagem gerada  $G$  ( $L(G)$ ):
  - Conjunto formado por todas as sentenças de símbolos terminais deriváveis a partir do símbolo inicial  $S$
  - $L(G) = \{s \mid s \text{ é um elemento de } T^* \text{ e } S \Rightarrow^+ s\}$
- Convenções
  - Símbolos que representam terminais em minúsculos:
    - $u, v, x, y, \dots$
  - 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:
    - $\alpha, \beta, \omega, \underline{w}, \underline{z}$

## Derivações

- Seja  $A \in N$ ,  $p = \{A \rightarrow \underline{v}\} \in P$ , e  $\underline{w}, \underline{z}$  quaisquer. Então:
  - $\underline{w}A\underline{z} \Rightarrow \underline{w}\underline{v}\underline{z}$
  - “derivação em um passo usando  $p$ ”
- Se  $\underline{w}_1 \Rightarrow \underline{w}_2 \Rightarrow \underline{w}_3 \dots \Rightarrow \underline{w}_n$  Então podemos dizer
  - $\underline{w}_1 \Rightarrow^* \underline{w}_n$
  - “derivação em múltiplos (0 ou mais) passos”; também  $\Rightarrow^*$  (um ou mais)

## Linguagem $L(G)$

- Como definir a Linguagem definida por  $G$  ?
  - $L(G)$ :
    - conjunto de strings  $\underline{w} \in T^*$  derivados a partir de  $S$ :
      - $S \Rightarrow^* \underline{w}$
      - $\underline{w}$  é uma sentença de  $G$
      - Forma Sentencial:  $S \Rightarrow^* \underline{v}$  e  $\underline{v}$  contém não-terminais
- Equivalência:
  - $G_1$  e  $G_2$  são equivalentes se  $L(G_1) = L(G_2)$

## Hierarquia de Chomsky

|        |                         |                                                                                             |
|--------|-------------------------|---------------------------------------------------------------------------------------------|
| Tipo 0 | Sem restrição Recursiva | Qualquer $\underline{u} \rightarrow \underline{v}$ desde que $\underline{u}$ seja não vazio |
| Tipo 1 | Sensível ao contexto    | $wA\underline{z} \rightarrow wvz$<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)                           |

13

## Hierarquia de Chomsky (II)

|        |                   |                                        |
|--------|-------------------|----------------------------------------|
| Tipo 0 |                   |                                        |
| Tipo 1 | Máquina de Turing |                                        |
| Tipo 2 | GLC               | Autômatos de Pilha Não Determinísticos |
| Tipo 3 | Regular           | AFND/AFD                               |

14

## Vericar se $L = L(G)$

- ⇒ Todos strings gerados estão em L  
 ⇐ Todos strings  $\underline{w} \in L$  podem ser gerados

Exemplo:  $T = \{a, b\}$

“todos strings com o mesmo número de ‘a’s e ‘b’s”

$$S \rightarrow \epsilon \mid a S b S \mid b S a S$$

15

## Prova

- ⇒ Todos strings gerados estão em L
- Prova por indução no tamanho do string:
    - Caso Base:  $\epsilon$  está em L
    - Passo de Indução: se  $\underline{u}, \underline{v}$  estão em L,  $|\underline{u}|, |\underline{v}| \leq n$ , provar que  $\underline{w}, |\underline{w}| > n$  está em L
    - Prova:
      - As construções possíveis:
        - $\underline{w} \Rightarrow \epsilon$  ou
        - $\underline{w} \Rightarrow a \underline{u} b \underline{v}$
        - $\underline{w} \Rightarrow b \underline{u} a \underline{v}$  (onde  $\underline{u}$  e  $\underline{v}$  são gerados por L)
      - Se  $\underline{u}$  e  $\underline{v}$  estão em L, então  $\underline{w}$  também está.
        - Dado que  $|\underline{u}| < |\underline{w}|$  e  $|\underline{v}| < |\underline{w}|$ , comprova-se pela hipótese

15

## Prova (II)

⇐ Todos strings  $\underline{w} \in L$  podem ser gerados

- Caso Base:  $\underline{w} = \epsilon$
- Passo de indução:
  - strings  $\underline{u}, \underline{v}, |\underline{u}|, |\underline{v}| \leq n$  podem ser gerados
- Prova:
  - Caso 1:  $\underline{w}$  começa com a
    - Encontre primeiro b a partir da direita de forma que  $w = a\underline{u}b\underline{v}$  e  $\underline{v}$  possui o mesmo número de a's e b's
    - Portanto  $\underline{u}, \underline{v} \in L$ . Desde que  $|\underline{u}| < |\underline{w}|$  e  $|\underline{v}| < |\underline{w}|$ , eles podem ser gerados pela hipótese
    - Derive  $\underline{w} \Rightarrow a \underline{u} b \underline{v}$
  - Caso 2:  $\underline{w}$  começa com b

17

## Análise Top-Down vs. Bottom Up

Gramática:  $S \rightarrow A B$

String: ccbca

$A \rightarrow c \mid \epsilon$

$B \rightarrow cbB \mid ca$

| Top-Down            |                     | Bottom-Up                |                     |
|---------------------|---------------------|--------------------------|---------------------|
| $S \Rightarrow AB$  | $S \rightarrow AB$  | $ccbca \Leftarrow Acbca$ | $A \rightarrow c$   |
| $\Rightarrow cB$    | $A \rightarrow c$   | $\Leftarrow AcbB$        | $B \rightarrow ca$  |
| $\Rightarrow ccbB$  | $B \rightarrow cbB$ | $\Leftarrow AB$          | $B \rightarrow cbB$ |
| $\Rightarrow ccbca$ | $B \rightarrow ca$  | $\Leftarrow S$           | $S \rightarrow AB$  |

18

## Árvore de derivação (Parse Tree)

- **Árvore de derivação:**

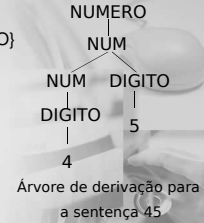
- Representação gráfica de uma derivação de sentença
- Estrutura hierárquica que originou a sentença
  - A raiz da árvore representa o símbolo inicial
  - Os vértices interiores são não-terminais
  - Os símbolos terminais e a palavra vazia são folhas

19

## Árvore de derivação – Exemplo 1

- Gramática  $G = (T, N, P, S)$

- $T = \{0, 1, 2, \dots, 9\}$
- $N = \{\text{NUMERO}, \text{NUM}, \text{DIGITO}\}$
- $P$ :
  - NUMERO  $\rightarrow$  NUM
  - NUM  $\rightarrow$  NUM DIGITO | DIGITO
  - DIGITO  $\rightarrow$  0|1|...|9
- $S$ : NUMERO



20

## Derivação mais à esquerda e mais à direita

- Derivação mais à esquerda de uma sentença:

- Seqüência de formas sentenciais que se obtém derivando sempre o símbolo **não-terminal** mais à esquerda.

- Derivação mais à direita de uma sentença:

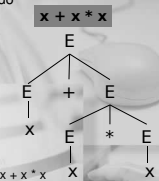
- Seqüência de formas sentenciais que se obtém derivando sempre o símbolo **não-terminal** mais à direita.

21

## Árvore de derivação – Exemplo 2

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

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



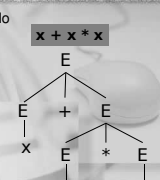
Possível derivação para a árvore:  
 $E \Rightarrow E + E \Rightarrow x + E \Rightarrow x + E * E \Rightarrow x + x * E \Rightarrow x + x * x$

A mesma árvore pode representar mais de uma derivação (à esquerda ou à direita) para uma mesma sentença.

## Árvore de derivação – Exemplo 2

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

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



Possível derivação para a árvore:

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

Outras possíveis derivações:

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

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

**Problemas:**

- 1- **baixo desempenho**, pois as derivações podem ser ambíguas
- 2- e se há uma **semântica** associada?

21

## Árvore de derivação – Exemplo 3

$E \rightarrow E \text{ Op } E \mid ( E ) \mid \text{Int}$

$\text{Op} \rightarrow * \mid / \mid + \mid -$

$\text{Int} \rightarrow [0-9]$

- Considere  $5 - 3 * 2$

24

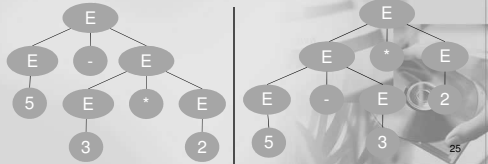
### Árvore de derivação – Exemplo 3

$E \rightarrow E \text{ Op } E \mid ( E ) \mid \text{Int}$

$\text{Op} \rightarrow * \mid / \mid + \mid -$

$\text{Int} \rightarrow [0-9]$

- Considere  $5 - 3 * 2$



### Outras Classificações de Gramáticas

- Gramática sem ciclos:
  - 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 do tipo  $A \rightarrow \epsilon$
  - Exceto produção  $S \rightarrow \epsilon$  ( $S$  é o símbolo inicial).

### Outras Classificações de Gramáticas

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

Reconhecedores top-down não aceitam gramáticas recursivas à esquerda

### Transformações de GLCs

- Eliminação de produções vazias
- Eliminação de recursividade à esquerda:
  - Recursão direta
  - Recursão indireta
- Fatoração de uma gramática

### Eliminação de produções vazias (1)

- Objetivo:
  - Eliminar produções da forma  $A \rightarrow \epsilon$ .
- Algoritmo: seja  $G = (T, N, P, S)$  uma GLC
  - Etapa 1:
    - Construir  $N_\epsilon$  conjunto de não-terminais que geram a palavra vazia:  $N_\epsilon = \{A \mid A \rightarrow \epsilon\}$ ;
    - 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.

### Eliminação de produções vazias (2)

- Etapa 2:
  - Construir conjunto de produções sem produções vazias.
  - Gera  $G_1 = (T, N, P_1, S)$ , onde  $P_1$  é construído como segue:
 
$$P_1 = \{A \rightarrow \alpha \mid \alpha \neq \epsilon\};$$
 Repita
    - Para todo  $A \rightarrow \alpha \in P_1$  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

### Eliminação de produções vazias (3)

– Etapa 3:

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

31

### Eliminação de recursão à esquerda

• Exemplo:

$A \rightarrow Aa \mid b$

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!

32

### Exemplo de eliminação de recursão à esquerda

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

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

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

$E \rightarrow TE'$   
 $E' \rightarrow +TE' \mid \epsilon$   
 $T \rightarrow FT'$   
 $T' \rightarrow ^*FT' \mid \epsilon$   
 $F \rightarrow (E) \mid Id$

33

### Convertendo recursão a esquerda para recursão à direita

Gramática original

$X \rightarrow Xb$   
 $\mid Xc$   
 $\mid AB$   
 $\mid C$

Introduzir Y

$X \rightarrow ABY$   
 $\mid CY$   
 $Y \rightarrow bY$   
 $\mid cY$   
 $\mid \epsilon$

34

### Fatoração de uma gramática

- Elimina 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:

$A \rightarrow \alpha X$

$X \rightarrow \beta_1 \mid \beta_2$

35

### Exemplo de Fatoração a Esquerda

Regras:

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

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

$Cmd \rightarrow \text{Outro}$

Fatorando à esquerda:

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

$Cmd \rightarrow \text{Outro}$

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

36

## Exercícios

- Exercícios 4.1 e 4.2 do livro do Dragão (Aho)
- Executar a seqüência de passos para transformação de GLC.
- Tendo como base a linguagem C, construa uma gramática para:
  - uma expressão aritmética, considerando operações básicas e parênteses.
  - uma expressão de atribuição.
  - uma expressão relacional/lógica, considerando operações básicas e parênteses.
  - o comando for.

37

## Bibliografia

### Leituras:

- A. M. A. Price, S. S. Toscani. **Implementação de Linguagens de Programação**: Compiladores. 3ª ed. Porto Alegre: Sagra-Luzzatto. 2005. Seção 3.1.
- A. V. Aho, R. Sethi, J. D. Ullman. **Compilers: Principles, Techniques and Tools**. Reading: Addison-Wesley. 1985. Seções 4.1, 4.2, 4.3. - Exercícios 4.1 e 4.2

### Bibliografia complementar:

- G. David Ripley, Frederick C. Druseikis. A Statistical Analysis of Syntax Errors. **Comput. Lang.** 3(4): 227-240 (1978).
- P. B. Menezes. **Linguagens Formais e Autômatos**. Porto Alegre: Sagra-Luzzatto. 2004.
- Todo material de Linguagens Formais.

38