

# **Teoria de Linguagens – *overview***

**(Prof. Marco Rodrigo Costa)**



# Conceitos básicos

- Alfabeto ( $\Sigma$ ): conjunto finito de símbolos
- Símbolo (ou caractere): entidade abstrata, não definida formalmente, como números, letras e outros
- Palavra (ou cadeia, ou sentença) ( $w$ ): seqüência finita de símbolos do alfabeto justapostos
  - Palavra vazia ( $\epsilon$ ): palavra sem símbolo
  - $\Sigma^*$ : conjunto de todas as palavras possíveis sobre  $\Sigma$
  - $\Sigma^+$ : conjunto de todas as palavras possíveis sobre  $\Sigma$  excetuando-se a palavra vazia  $\Rightarrow \Sigma^+ = \Sigma^* - \epsilon$

# Linguagem Formal

- Linguagem formal: é um conjunto de palavras sobre um alfabeto
  - Exemplo: considere o alfabeto  $\Sigma = \{a, b\}$ . São exemplos de linguagens sobre  $\Sigma$ :
    - Conjunto vazio:  $= \{ \}$
    - Conjunto formado pela palavra vazia  $= \{ \varepsilon \}$
    - Conjunto de palíndromos (palavras que têm a mesma leitura da esquerda para a direita):  $\{\varepsilon, a, b, aa, bb, aba, aaaa, \dots\} \Rightarrow$  linguagem infinita

# Gramática

- Gramática: é uma quádrupla ordenada  $G = (V, T, P, S)$ , onde:
  - $V$  é um conjunto finito de símbolos **variáveis** ou **não-terminais**
  - $T$  é um conjunto finito de símbolos **terminais** disjunto de  $V$
  - $P$  é um conjunto finito de pares, denominados **regras de produção**, onde a primeira componente é palavra de  $(V \cup T)^+$  e a segunda componente é palavra de  $(V \cup T)^*$
  - $S$  é um elemento de  $V$  denominado **variável inicial**
- Uma regra de produção  $(\alpha, \beta)$  é representada por  $\alpha \rightarrow \beta$
- As regras de produção definem as condições de geração das palavras da linguagem  $\Rightarrow$  gramática é formalismo de geração
- $\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, \dots, \alpha \rightarrow \beta_n \equiv \alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$

# Gramática...

- Derivação: seja  $G = (V, T, P, S)$ , uma gramática. Uma derivação é um par da relação denotada por  $\Rightarrow$  com domínio em  $(V \cup T)^+$  e contra-domínio em  $(V \cup T)^*$ . Logo, uma derivação  $(\alpha, \beta)$  é representada por  $\alpha \Rightarrow \beta$ 
  - A derivação é a substituição de uma subpalavra de acordo com uma regra de produção

# Gramática...

- Linguagem gerada: seja  $G = (V, T, P, S)$ , uma gramática. A linguagem gerada pela gramática  $G$ , denotada por  $L(G)$ , é composta por todas as palavras de símbolos terminais deriváveis a partir do símbolo inicial  $S$ , ou seja:  
$$L(G) = \{w \in T^* \mid S \Rightarrow^+ w\}$$
- Gramáticas equivalentes: duas gramáticas  $G_1$  e  $G_2$  são ditas equivalentes se, e somente se,  $L(G_1) = L(G_2)$ .

# Gramática...

- Convenções: utiliza-se, para:
  - Símbolos variáveis:  $A, B, \dots, S, T$
  - Símbolos terminais:  $a, b, \dots, s, t$
  - Palavras de símbolos terminais:  $u, v, w, x, y, z$
  - Palavras de símbolos terminais ou variáveis:  $\alpha, \beta, \dots$
- Exemplos (gramática, derivação e linguagem gerada):
  - $V = \{S, A, B\}, T = \{a, b\}, P = \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\}$
  - <Exemplo 22, pgs. 24 e 25, ref. Menezes>

# Sistema de Estados Finitos

- Sistema de Estados Finitos: é um modelo matemático de sistema com entradas e saídas discretas
- Pode assumir um número finito e pré-definido de estados
- Cada estado resume somente as informações do passado necessárias para determinar as ações para a próxima entrada
- Exemplo clássico: elevador. Não memoriza as requisições anteriores. Cada “estado” sumariza as informações “andar corrente” e “direção de movimento”. As entradas são as requisições pendentes
- Exemplos: autômatos finitos (determinísticos ou não), autômatos com pilha e máquinas de Turing



# Linguagens Formais – Contextos

- Teoria de linguagens
  - Projetos de Linguagens de Programação
  - Reconhecedores e tradutores de Linguagens de Programação. Especificamente, por exemplo, nas fases de análise léxica e sintática
- Implementação de linguagens
  - Empregada em técnicas de desenvolvimento de compiladores
- Problemas computacionais
  - Útil no estudo de viabilidade computacional de soluções de problemas
- Conceitos de linguagens de programação
  - Útil para melhor programação e projetos de linguagens