# ACH2043
# INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

## Aula 6
## Últimas observações sobre autômatos e Cap. 2.1 – Gramáticas Livres de Contexto

Profa. Ariane Machado Lima
ariane.machado@usp.br

# Últimas observações sobre autômatos

- Máquinas de Mealy: aceitam símbolos na transição (autômatos tradicionais)
- Máquinas de Moore: aceitam símbolos nos estados (HMM – modelo oculto de Markóv)
- Transdutores: geram uma cadeia de saída

# Transdutor finito do tipo Máquina de Mealy

- $T_{Mealy} = (Q, \Sigma, \Delta, \delta, \lambda, q_0, F)$ sobre um autômato finito $M = (Q, \Sigma, \delta, q_0, F)$
  - $\Delta$ é o alfabeto de saída
  - $\lambda: Q \times \Sigma \rightarrow \Delta^*$ é a função de transdução

# Transdutor finito do tipo Máquina de Mealy

- Exemplo: $T_{Mealy} = (Q, \Sigma, \Delta, \delta, \lambda, q_0, F)$ onde:
  - $Q = \{q_0, q_1\}$
  - $\Sigma = \{a, b, c\}$
  - $\Delta = \{a, b, c\}$
  - $\delta = \{(q_0,a) \rightarrow q_1, (q_1,b) \rightarrow q_1, (q_1,c) \rightarrow q_0\}$
  - $\lambda = \{(q_0,a) \rightarrow ab, (q_1,b) \rightarrow \varepsilon, (q_1,c) \rightarrow c\}$
  - $F = \{q_1\}$

# Transdutor finito do tipo Máquina de Moore

- $T_{Moore} = (Q, \Sigma, \Delta, \delta, \lambda, q_0, F)$ sobre um autômato finito $M = (Q, \Sigma, \delta, q_0, F)$
  - $\Delta$ é o alfabeto de saída
  - $\lambda: Q \rightarrow \Delta^*$ é a função de transdução

# Transdutor finito do tipo Máquina de Moore

- Exemplo: $T_{Moore} = (Q, \Sigma, \Delta, \delta, \lambda, q_0, F)$ onde:
  - $Q = \{q_0, q_1\}$
  - $\Sigma = \{a, b, c\}$
  - $\Delta = \{1\}$
  - $\delta = \{(q_0,a) \rightarrow q_1, (q_1,b) \rightarrow q_1, (q_1,c) \rightarrow q_0\}$
  - $\lambda = \{q_0 \rightarrow 1, q_1 \rightarrow \varepsilon\}$
  - $F = \{q_1\}$

# Equivalência dos transdutores

- Teorema: Toda Máquina de Mealy pode ser simulada por uma Máquina de Moore, e vice-versa.

# Minimização de autômatos finitos

- Vários autômatos podem gerar a mesma linguagem
- Cada linguagem regular é reconhecida por um autômato finito determinístico mínimo (com relação ao número de estados) e único
  - Utilidades:
    - Gerar um reconhecedor o mais compacto e eficiente possível
    - Comparar se duas linguagens são equivalentes

# Minimização de autômatos finitos

- Processo:
  - Eliminação de estados inacessíveis
    - Não há caminho de $q_0$ até ele
  - Eliminação de estados inúteis
    - Não conduzem a um estado final
  - Agrupamento e fusão de estados equivalentes
- Ex:
  - $\delta = \{(q_0, a) \rightarrow q_3, (q_0, a) \rightarrow q_4, (q_0, b) \rightarrow q_2, (q_1, c) \rightarrow q_3\}$
  - $F = \{q_3, q_4\}$

# Perigo das transições no vazio

- M = …
  - $\delta = \{(q_0, \varepsilon) \rightarrow q_1, (q_0, a) \rightarrow q_2, (q_1, \varepsilon) \rightarrow q_0, (q_2, b) \rightarrow q_3\}$
  - $F = \{q_1, q_3\}$

# Perigo das transições no vazio

- M = …
  - $\delta = \{(q_0, \varepsilon) \rightarrow q_1, (q_0, a) \rightarrow q_2, (q_1, \varepsilon) \rightarrow q_0, (q_2, b) \rightarrow q_3\}$
  - $F = \{q_1, q_3\}$

- O autômato não pára!

- Felizmente há um algoritmo para eliminação de transições no vazio

# Referências (complementares)

RAMOS, M. V. M.; NETO, J. J.; VEGA, I. S. **Linguagens Formais**. Ed. Bookman,

- HMM:

  RABINER, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. **Proceedings of the IEEE**, v. 77, n. 2, p. 257-286 1989

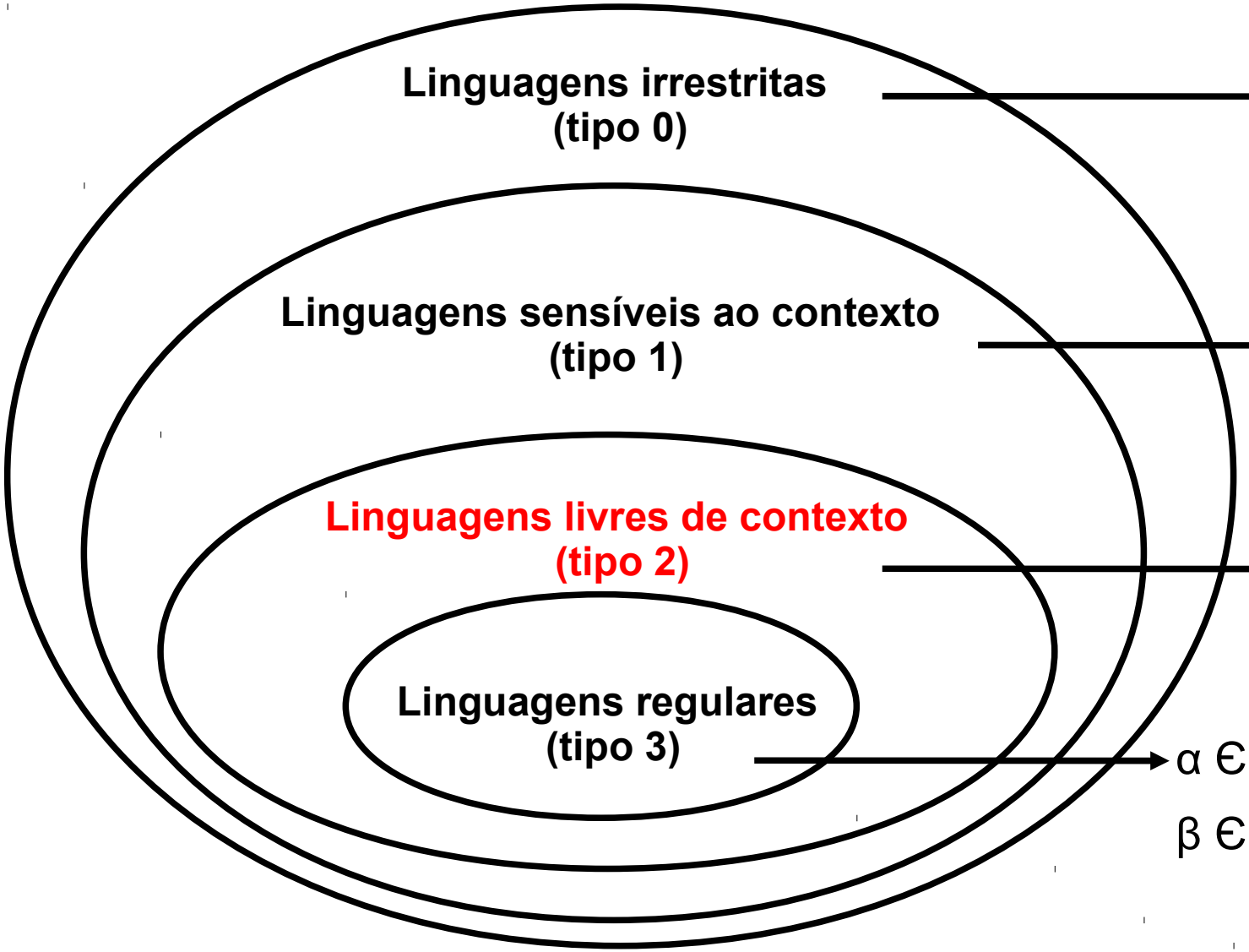
# Cap 2 – Linguagens livres de contexto

- 2.1 – Gramáticas Livres de Contexto
- 2.2 – Autômato com Pilha
- 2.3 – Linguagens não-livres de contexto

# 2.1 – Gramáticas Livres de Contexto

# Hierarquia de Chomsky

$\alpha \rightarrow \beta$




$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$

$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$
$|\alpha| <= |\beta|$

$\alpha \in V$

$\beta \in (V \cup \Sigma)^*$

$\alpha \in V$

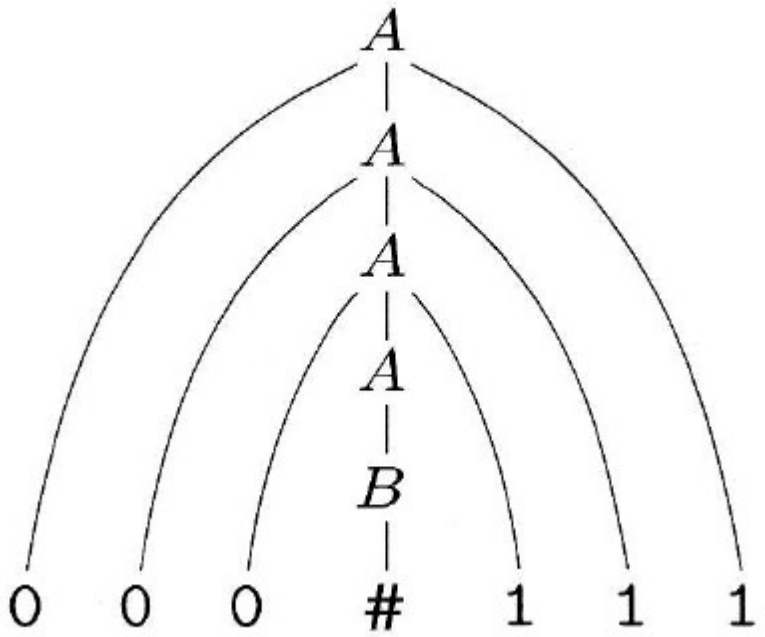
$\beta \in \Sigma_\varepsilon$, $\beta \in V$, $\beta \in (V\Sigma$ ou $\Sigma V)$

# Gramáticas Livres de Contexto

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

Por exemplo, a gramática $G_1$ gera a cadeia 000#111.

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$




Árvore sintática
ou
Árvore de derivação

# Dicas para projetar gramáticas livres de contexto

- É a união de linguagens mais simples?

Por exemplo, para obter uma gramática para a linguagem $\{0^n1^n \mid n \geq 0\} \cup \{1^n0^n \mid n \geq 0\}$, primeiro construa a gramática

$$S_1 \rightarrow 0S_11 \mid \varepsilon$$

para a linguagem $\{0^n1^n \mid n \geq 0\}$ e a gramática

$$S_2 \rightarrow 1S_20 \mid \varepsilon$$

para a linguagem $\{1^n0^n \mid n \geq 0\}$ e então adicione a regra $S \rightarrow S_1 \mid S_2$ para dar a gramática

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \\ S_1 &\rightarrow 0S_11 \mid \varepsilon \\ S_2 &\rightarrow 1S_20 \mid \varepsilon \,. \end{aligned}$$

# Dicas para projetar gramáticas livres de contexto

- Definições recursivas

**EXEMPLO 2.3**

Considere a gramática $G_3 = (\{S\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$. O conjunto de regras, $R$, é

$$S \to \mathtt{a}S\mathtt{b} \mid SS \mid \varepsilon.$$