Teoria da Computação

Gramáticas

versão 1.1

Prof. D.Sc. Fabiano Oliveira
fabiano.oliveira@ime.uerj.br

Gramáticas

- Uma ***gramática*** G é uma tupla <N, Σ, P, S> com vocabulário $V = N \cup \Sigma$, onde:
  - N é um alfabeto de símbolos auxiliares, chamados de ***símbolos não-terminais***
  - Σ é o alfabeto no qual a linguagem é definida, cujos símbolos são chamados de **terminais**
  - $P \subseteq V^* \times V^*$ é o ***conjunto de regras de reescrita***
  - $S \in N$ é o ***símbolo inicial***

Gramáticas

- Uma ***gramática*** G é uma tupla <N, Σ, P, S> com vocabulário V = N ∪ Σ, onde:

  Ex.:

  $G_1 = < \{ S \}, \{ 0, 1 \}, \{ (S, 0S1), (S, \varepsilon) \}, S >$

  $G_2 = < \{ S, T \}, \{ a \},$
  $\{ (S, aaS), (S, aT), (T, \varepsilon) \}, T >$

Gramáticas

- **Notação**: $(\alpha, \beta_1), (\alpha, \beta_2), \ldots, (\alpha, \beta_n) \in P$ então escrevemos $\alpha \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

  Ex.:

  $G_1 = < \{ S \}, \{ 0, 1 \}, \{ S \rightarrow 0S1 \mid \varepsilon \}, S >$

  $G_2 = < \{ S, T \}, \{ a \}, \{ S \rightarrow aaS \mid aT, T \rightarrow \varepsilon \}, T >$

Gramáticas

- **Notação**: se houver a convenção de que:
  - os não-terminais serão denotados por letras maiúsculas
  - os terminais por números e letras minúsculas
  - todos os símbolos terminais e não-terminais aparecem em alguma regra de reescrita
  - o símbolo inicial é o S

  então, a gramática fica definida apenas pelas regras de reescrita.

  Ex.:
  $G = < \{ S \}, \{ 0, 1 \}, \{ S \rightarrow 0S1 \mid \varepsilon \}, S >$, ou
  $G = S \rightarrow 0S1 \mid \varepsilon$

Gramáticas

- Dado gramática $G = \langle N, \Sigma, P, S \rangle$, a relação de $V^* \times V^*$ chamada de ***derivação*** e denotada por $\Rightarrow$ é aquela tal que:

$$\gamma\alpha\delta \Rightarrow \gamma\beta\delta \Leftrightarrow \alpha \rightarrow \beta \in P, \text{ onde } \gamma, \delta \in V^*$$

Ex.: $G_1 = \langle \{ S \}, \{ 0, 1 \}, \{ S \rightarrow 0S1 \mid \varepsilon \}, S \rangle$

$S \Rightarrow 0S1$, $000S1 \Rightarrow 0000S11$, $0000S \Rightarrow 0000$,

$(0S1, 000S111) \notin \Rightarrow$

Gramáticas

- Ex.: $G_1 = S \to 0S1 \mid \varepsilon$

  $S \Rightarrow^2 00S11$

  $(S, 00001111) \notin \Rightarrow$

  $(S, 00001111) \notin \Rightarrow^2$

  $(S, 00001111) \notin \Rightarrow^3$

  $S \Rightarrow^4 00001111$

  $\therefore S \Rightarrow^* 00001111$

  $(S, 0001111) \notin \Rightarrow^*$

Gramáticas

- A ***linguagem de uma gramática*** G = <N, Σ, P, S> é a linguagem $L(G) = \{ x \in \Sigma^* \mid S \Rightarrow^* x \}$

  $G_1 = S \rightarrow 0S1 \mid \varepsilon$
  $L(G_1) = \{ 0^i 1^i : i \in \mathbb{N} \}$

  $G_2 = S \rightarrow aaS \mid aT,\ T \rightarrow \varepsilon$
  $L(G_2) = \{ a^{2i+1} : i \in \mathbb{N} \}$

Gramáticas

- Duas gramáticas G_1 e G_2 são ***equivalentes*** se $L(G_1) = L(G_2)$

  Ex.:
  $G_1 = S \rightarrow aTa \mid bTb,\ T \rightarrow aT \mid bT \mid \varepsilon$
  $G_2 = S \rightarrow aA \mid bB,\ A \rightarrow aA \mid bA \mid a,\ B \rightarrow aB \mid bB \mid b$

  $L(G_1) = L(G_2) = ?$

Gramáticas

- **Hierarquia de Gramáticas**
  - **Tipo 0 (sem restrições, gsr)**: gramática geral
  - **Tipo 1 (sensível ao contexto, gsc)**: regras $\alpha \rightarrow \beta$ tais que $|\alpha| \leq |\beta|$ ou ($S \rightarrow \varepsilon$ se S não aparece do lado direito de nenhuma regra)
  - **Tipo 2 (livre de contexto, glc)**: regras na forma $A \rightarrow \beta$ tal que $A \in N$
  - **Tipo 3 (regular, gr)**. cada regra na forma:
    - $A \rightarrow aB$ tal que $A, B \in N$
    - $A \rightarrow a$, $A \in N$, $a \in \Sigma$
    - $A \rightarrow \varepsilon$, $A \in N$

Gramáticas

- **Hierarquia de Gramáticas**

  Ex:

  - $G_1 = S \rightarrow aTa \mid bTb, T \rightarrow aT \mid bT \mid \varepsilon$ **é** gsr, **não é** gsc, **é** glc, **não é** gr

  - $G_2 = S \rightarrow aA \mid bB, A \rightarrow aA \mid bA \mid a, B \rightarrow aB \mid bB \mid b$ **é** gsr, **é** gsc, **é** glc, **é** gr

Gramáticas

- Uma ***linguagem*** L ***é regular (lr)/livre de contexto (llc)/sensível ao contexto (lsc)/sem restrições (lsr)*** se existe uma gramática G deste respectivo tipo tal que L(G) = L

  Ex:

  - L(S → aTa | bTb, T → aT | bT | ε) **é** lsr, (**não é** lsc??), **é** llc, (**não é** lr??)

  - L(S → aA | bB, A → aA | bA | a, B → aB | bB | b) **é** lsr, **é** lsc, **é** llc, **é** lr

Gramáticas

Teorema:
Para qualquer glc G, existe uma glc G' equivalente a G que satisfaz a definição de gsc.

Gramáticas

- **Hierarquia de Gramáticas e Linguagens**

Regular
$A \to aB$, $A \to a$, $A \to \varepsilon$

$\subset$

Livre de Contexto
$A \to \beta$

não $\subset$

Sensível ao Contexto $|\alpha| \leq |\beta|$ ou ($S \to \varepsilon$ se S não aparece do lado direito)

$\subset$

Sem restrições
$\alpha \to \beta$ quaisquer

Hierarquia das **Gramáticas**

Regular

$\subset$

Livre de Contexto

$\subset$ (teorema anterior!)

Sensível ao Contexto

$\subset$

Sem restrições

Hierarquia das **Linguagens**

Gramáticas

- **Exercício**: Mostre que a linguagem $\{ a^i b c^j : i, j \in \mathbb{N} \}$ é lr.

Gramáticas

- **Exercício**: Mostre que a linguagem $\{ a^i b c^i : i \in \mathbb{N} \}$ é llc.

  (**Nota**: Veremos mais tarde como argumentar que tal linguagem **não é** lr).

Gramáticas

- **Exercício**: Mostre que a linguagem { xx : x ∈ {a, b}* } é lsc.

  (**Nota**: Veremos mais tarde como argumentar que tal linguagem **não é** llc).

Gramáticas

Teorema:
Toda lsc é recursiva.

∴ Se existe uma gsc que gera uma linguagem L, existe um procedimento que em tempo finito determina se uma cadeia de entrada pertence ou não a L!

Gramáticas

Dem.: Sejam G = <N, Σ, P, S> uma gsc, e L = L(G). Uma gsc pode ter uma regra $S \rightarrow \varepsilon$, se S não aparecer à direita em nenhuma regra. Para cada uma das outras regras, o comprimento do lado esquerdo não pode ser maior que o do lado direito, e portanto cada aplicação da regra pode manter ou aumentar o comprimento da cadeia sendo derivada, mas nunca diminuí-la. Portanto, se $x \neq \varepsilon$, $S \Rightarrow^* \gamma \Rightarrow^* x$ implica que $1 \leq |\gamma| \leq |x|$. Seja o algoritmo:

função ReconheceG ($x \in \Sigma^*$): Lógico

1. **se** $x \neq \varepsilon$, **vá para** 3.
2. **se** $S \rightarrow \varepsilon \in P$ **então retornar** (V), **senão retornar** (F)
3. $X \leftarrow \{ S \}$, $Y \leftarrow \varnothing$. //X: cadeias a derivar; Y: cadeias já derivadas
4. **se** $x \in X$ **então retornar** (V)
5. **se** $X = \varnothing$, **então retornar** (F)
6. Tome $\alpha \in X$ e faça $X \leftarrow X \setminus \{\alpha\}$; $Y \leftarrow Y \cup \{\alpha\}$
7. para cada β tal que $\alpha \Rightarrow \beta$, **se** $\beta \notin Y$ e $|\beta| \leq |x|$ **então** $X \leftarrow X \cup \{\beta\}$
8. **vá para** 4

Gramáticas

Teorema:
Toda lsr é um conjunto recursivamente enumerável..

∴ Se existe uma gramática que gera uma linguagem L, existe um procedimento que enumera cada cadeia de L em tempo finito!

Gramáticas

Dem.: Seja $G = \langle N, \Sigma, P, S \rangle$ uma gramática. O procedimento abaixo enumera os elementos de $L(G)$.

procedimento EnumeraG ()

1. $X \leftarrow \{ S \}$ //X: cadeias a derivar
2. **para cada** $\alpha \in X \cap \Sigma^*$ **faça escrever** (α)
3. $X \leftarrow \{ \beta \in V^* \mid \alpha \in X \text{ e } \alpha \Rightarrow \beta \}$
4. **vá para** 2

A cada iteração do procedimento acima, X contém todas as sequências deriváveis de S em um passo adicional. Assim, se $S \Rightarrow^* x$ em n passos, em n passos o procedimento escreverá x.