Engenharia de Computação

ECM253 – Linguagens Formais, Autômatos e Compiladores

Modelos de Computação – Linguagens e Gramáticas

Slides da disciplina ECM253 – Linguagens Formais, Autômatos e Compiladores
Curso de Engenharia de Computação
Instituto Mauá de Tecnologia
Prof. Marco Antonio Furlan de Souza

Linguagens e computação

- Uma **linguagem** L é um **conjunto** (finito ou infinito) de **cadeias** sobre um alfabeto Σ, $L \subseteq \Sigma^*$;
- Para uma **definição computacional de uma linguagem**, pode-se especificar:
  - **Um gerador da linguagem**: que enumere os elementos da linguagem – **gramática** ou
  - **Um reconhecedor da linguagem**: que decida se uma cadeia está ou não na linguagem, retornando `V` se estiver e `F`, caso contrário – **autômato**.

- **Exemplos de linguagens**
  - Seja $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. $\Sigma^* = \{\epsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{aaa}, \mathtt{aab}, \ldots\}$. São exemplos de linguagens sobre Σ: $\emptyset, \{\epsilon\}, \{\mathtt{a}, \mathtt{b}\}, \{\epsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \mathtt{aaaaa}\}$.
  - Quando presentes, todos os $\mathtt{b}$'s são precedidos por todos $\mathtt{a}$'s:

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : \text{ quando presentes, } \mathtt{a}\text{'s precedem os } \mathtt{b}\text{'s em } w\}$$
$$= \{\epsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aabbb}, \mathtt{bbb}, \ldots\}$$

  - Cadeias que terminam em $\mathtt{a}$:

$$L = \{x : \exists y \in \{\mathtt{a}, \mathtt{b}\}^* (x = y\mathtt{a})\} = \{\mathtt{a}, \mathtt{aa}, \mathtt{bbaa}, \mathtt{ba}, \ldots\}$$

  - Linguagem vazia (não contém cadeia alguma):

$$L = \{\} = \emptyset$$

  - Linguagem contendo apenas a cadeia vazia:

$$L = \{\epsilon\}$$

- **Exemplos de linguagens**
  - Nenhum prefixo contém b:

$$L = \{w \in \{\mathtt{a},\mathtt{b}\}^* : w \text{ não contenha } \mathtt{b} \text{ em seu prefixo}\}$$
$$= \{\epsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \mathtt{aaaaa}, \mathtt{aaaaaa}, \ldots\}$$

  - Nenhum prefixo começa com b:

$$L = \{w \in \{\mathtt{a},\mathtt{b}\}^* : \text{o prefixo de } w \text{ não se inicie por } \mathtt{b}\}$$
$$= \{\epsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aba}, \mathtt{aaabb}, \mathtt{abbaa}, \mathtt{abbbabaa}, \ldots\}$$

  - Todo prefixo começa com b:

$$L = \{w \in \{\mathtt{a},\mathtt{b}\}^* : \text{toda cadeia } w \in \{\mathtt{a},\mathtt{b}\}^* \text{ possui prefixo iniciado por } \mathtt{b}\}$$
$$= \emptyset$$

- **Exemplos de linguagens**
  - Uso de replicação para definir uma linguagem:

$$L = \{\mathtt{a}^n : n \geq 0\}$$
$$= \{\epsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \ldots\}$$

  - Exemplo já apresentado, reescrito:

$$L = \{\mathtt{a}^m\mathtt{b}^n : m, n \geq 0\}$$
$$= \{\epsilon, \mathtt{a}, \mathtt{aa}, \mathtt{aabbb}, \mathtt{bbb}, \ldots\}$$

- **Cardinalidade de uma linguagem**

Teorema

Se $\Sigma \neq \emptyset$ então Σ^* é contavelmente infinita.

Demonstração.

Os elementos de Σ^* podem ser enumerados lexicograficamente pelo procedimento a seguir:

- Enumerar todas as cadeias de tamanho zero, depois de tamanho um, de tamanho dois etc.;
- Enumerar as cadeias de mesmo tamanho de acordo com a ordem de dicionário;

Esta enumeração é infinita pois não existe uma cadeia mais longa em Σ^*. Assim, trata-se de uma enumeração infinita. □

- **Quantas linguagens existem?**

Teorema

Existe um número incontavelmente infinito de linguagens.

Demonstração.

O conjunto de linguagens definida por Σ é $\wp(\Sigma^*)$. Sabe-se que o conjunto Σ^* é contavelmente infinito, mas o conjunto $\wp(\Sigma^*)$ não é contavelmente infinito pois pode-se, a partir de uma tentativa de enumeração de $\wp(\Sigma^*)$, determinar um novo e inédito elemento não previsto anteriormente, e assim por diante, contradizendo a proposta de um conjunto enumerado. A técnica utilizada para esta prova é a **diagonalização de Cantor**. □

- **Operações de conjuntos aplicadas às linguagens**
  - Como uma linguagem é um conjunto, todas as operações sobre conjuntos aplicam-se às linguagens. Seja, por exemplo:

$$\Sigma = \{\mathtt{a}, \mathtt{b}\}$$

$$L_1 = \{\text{cadeias com um número par de } \mathtt{a}\text{'s}\}$$

$$L_2 = \{\text{cadeias sem } \mathtt{b}\text{'s}\}$$

Então:

$$L_1 \cup L_2 = \{\text{todas as cadeias com apenas } \mathtt{a}\text{'s e cadeias contendo } \mathtt{b}\text{'s com um número par de } \mathtt{a}\text{'s}\}$$

$$L_1 \cap L_2 = \{\epsilon, \mathtt{aa}, \mathtt{aaaa}, \mathtt{aaaaaa}, \ldots\}$$

$$L_2 - L_1 = \{\mathtt{a}, \mathtt{aaa}, \mathtt{aaaaa}, \ldots\}$$

$$\neg(L_2 - L_1) = \{\text{cadeias com no mínimo um } \mathtt{b}\} \cup \{\text{cadeias com número par de } \mathtt{a}\text{'s}\}$$

- **Concatenação de linguagens**
  - Sejam L_1 e L_2 duas linguagens definidas sobre algum alfabeto Σ. Sua **concatenação**, L_1L_2 é:

$$L_1L_2 = \{w \in \Sigma^* : \exists s \in L_1(\exists t \in L_2(w = st))\}$$

- Exemplo:

$$\begin{aligned} L_1 &= \{\texttt{cat}, \texttt{dog}, \texttt{mouse}, \texttt{bird}\} \\ L_2 &= \{\texttt{bone}, \texttt{food}\} \\ L_1L_2 &= \{\texttt{catbone}, \texttt{catfood}, \texttt{dogbone}, \texttt{dogfood}, \texttt{mousebone}, \\ &\quad \texttt{mousefood}, \texttt{birdbone}, \texttt{birdfood}\} \end{aligned}$$

- **Fecho de Kleene**
  - Seja L uma linguagem sobre algum alfabeto Σ. O **fecho de Kleene** de L, L^* é:

$$L^* = \{\epsilon\} \cup \{w \in \Sigma^* : \exists k \geq 1 (\exists w_1, w_2, \ldots, w_k \in L (w = w_1 w_2 \ldots w_k))\}$$

  - **Exemplo**:

$$L = \{\texttt{cat}, \texttt{dog}, \texttt{fish}\}$$
$$L^* = \{\epsilon, \texttt{dog}, \texttt{cat}, \texttt{fish}, \texttt{dogdog}, \texttt{dogcat}, \ldots, \texttt{fishdog}, \ldots, \texttt{fishcatfish}, \texttt{fishdogfishcat}, \ldots\}$$

  - L^* sempre conterá um **número infinito de cadeias** desde que L não seja igual à $\emptyset$ ou $\{\epsilon\}$;
  - Se L^* deve conter pelo menos um elemento de L, define-se: $L^+ = LL^*$.

- **Inversa de uma linguagem**
  - Seja L uma linguagem sobre algum alfabeto Σ. A **inversa** de L, L^R é:

$$L^R = \{w \in \Sigma^* : w = x^R \text{ para algum } x \in L\}$$

  - Basta inverter as cadeias de L.

Teorema

Se L_1 e L_2 são linguagens, então $(L_1L_2)^R = L_2^R L_1^R$.

Demonstração.

$$\begin{aligned}(L_1L_2)^R &= \{(xy)^R : x \in L_1 \wedge y \in L_2\} \\ &= \{y^R x^R : x \in L_1 \wedge y \in L_2\} \\ &= L_2^R L_1^R\end{aligned}$$

□

- **Conceitos**
  - Uma **gramática formal** G é uma **forma matemática precisa e compacta** de **definição** de uma **linguagem** L;
  - Assim, **dispensa a listagem** de **todas as cadeias legais** de uma linguagem;
  - Implica em um **algoritmo** que **permite gerar todas as cadeias legais** da linguagem, **comumente** de **modo recursivo**;
  - Uma forma conhecida para **descrever gramáticas** recursivamente é a **gramática de estrutura frasal**, elaborada por **Noam Chomsky** na década de 1950.

- **Gramática de estrutura frasal**

Definição

Uma gramática de estrutura frasal, $G = (V, \Sigma, R, S)$ é uma quádrupla na qual:

- V é o **alfabeto** (ou vocabulário) da **linguagem** e contém símbolos (metasímbolos) que são utilizados apenas na definição da linguagem denominados de **não-terminais** (ou **variáveis da gramática**) e também símbolos **terminais**, que aparecem apenas nas cadeias da linguagem;
- Σ é o conjunto de **símbolos terminais**, $\Sigma \subset V$;
- R é um **conjunto** não vazio **de regras de produção** ou **regras de reescrita**, assim definido: $R \subset (V^* \times (V - \Sigma) \times V^*) \times V^*$, cujos pares (α, β) são escritos como $\alpha \rightarrow \beta$;
- S é o símbolo não-terminal **de partida** ou **inicial**, $S \in V - \Sigma$.

- **Exemplo**. A gramática $G = (V, \Sigma, R, S)$, a seguir, gera uma linguagem de identificadores de uma linguagem de programação:

$$
\begin{aligned}
V = &\{S, I, L, D, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}, \mathtt{k}, \mathtt{l}, \mathtt{m}, \mathtt{n}, \mathtt{o}, \mathtt{p}, \mathtt{q}, \mathtt{r}, \mathtt{s}, \mathtt{t}, \\
&\mathtt{u}, \mathtt{v}, \mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{z}, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
\Sigma = &\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}, \mathtt{k}, \mathtt{l}, \mathtt{m}, \mathtt{n}, \mathtt{o}, \mathtt{p}, \mathtt{q}, \mathtt{r}, \mathtt{s}, \mathtt{t}, \\
&\mathtt{u}, \mathtt{v}, \mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{z}, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
R = &\{S \to I, I \to L, I \to IL, I \to ID, \\
&L \to \mathtt{a}, L \to \mathtt{b}, \ldots, L \to \mathtt{z}, \\
&D \to 0, D \to 1, \ldots, D \to 9\}
\end{aligned}
$$

- **Derivação direta**

Definição

Seja $G = (V, \Sigma, R, S)$ uma gramática. Para $\sigma, \psi \in V^*$, σ é dito **diretamente derivável** de ψ, escrito como $\psi \Rightarrow \sigma$, se existem cadeias ϕ_1 e ϕ_2 (incluindo cadeias vazias) tais que $\psi = \phi_1 \alpha \phi_2$ e $\sigma = \phi_1 \beta \phi_2$ e $\alpha \rightarrow \beta$ é uma produção de G. Quando $\psi \Rightarrow \sigma$ diz-se que ψ produz diretamente σ ou ainda que σ reduz-se diretamente à ψ.

- **Derivação de cadeia**

Definição

Seja $G = (V, \Sigma, R, S)$ uma gramática. A cadeia ψ **produz** σ (ou σ reduz-se à ψ), escrita como $\psi \overset{*}{\Rightarrow} \sigma$ se existem cadeias $\phi_0, \phi_1, \ldots, \phi_n$, com $n \geq 0$, tais que $\psi = \phi_0 \Rightarrow \phi_1, \phi_1 \Rightarrow \phi_2, \ldots, \phi_{n-1} \Rightarrow \phi_n = \sigma$. A relação $\overset{*}{\Rightarrow}$ é o fecho transitivo reflexivo da relação $\Rightarrow$.

- **Exemplo**
  - Na gramática apresentada anteriormente, o símbolo de início é S. É a partir de uma produção em que ele aparece do lado esquerdo que se começa qualquer derivação;
  - Assim, uma derivação de cadeia válida é a seguinte:

$$S \Rightarrow I \Rightarrow ID \Rightarrow IDD \Rightarrow LDD \Rightarrow \texttt{a}DD \Rightarrow \texttt{a1}D \Rightarrow \texttt{a13}$$

  - Então, $S \overset{*}{\Rightarrow} \texttt{a13}$.
  - Esta gramática permite derivar `23q`? e `r2d2`?

Linguagem gerada por uma gramática

- **Processo de geração de uma linguagem**
  - Uma **forma sentencial** é qualquer cadeia derivada a partir do não-terminal S;
  - A **linguagem** L **gerada** por uma gramática G é o conjunto de todas as formas sentenciais cujos símbolos são **terminais**:

$$L(G) = \{\sigma : S \overset{*}{\Rightarrow} \sigma (\sigma \in \Sigma^*)\}$$

  - Em outras palavras: uma **linguagem é um subconjunto** do **conjunto de todas as cadeias** (sentenças) **terminais** sobre Σ^*.

Teste seus conhecimentos

(1) Elaborar uma gramática de estrutura frasal que gere cadeias para a linguagem $L = (11)^*0$, o conjunto de todas as cadeias consistindo de algum número de concatenações de 11 com ele próprio, seguido finalmente por 0.

(2) Elaborar uma gramática de estrutura frasal que gere a linguagem $L = \{0^n 1^n : n \in \mathbb{N}\}$.

(3) Seja $V = \{S, A, B, \mathtt{a}, \mathtt{b}\}$ e $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Descobrir a linguagem gerada pela gramática $G = (V, \Sigma, R, S)$ quando:

a) $R = \{S \to AB, A \to \mathtt{ab}, B \to \mathtt{bb}\}$.

b) $R = \{S \to AB, S \to AA, A \to \mathtt{a}B, A \to \mathtt{ab}, B \to \mathtt{b}\}$.

c) $R = \{S \to AB, A \to \mathtt{a}B\mathtt{b}, B \to \mathtt{b}B\mathtt{a}, A \to \epsilon, B \to \epsilon\}$.

Teste seus conhecimentos

(4) Elaborar uma uma gramática de estrutura frasal para todas as cadeias binárias contendo o símbolo 1 seguido de um número impar de 0's.

(5) Elaborar uma uma gramática de estrutura frasal para todas as cadeias binárias contendo um número de símbolos 1's diferente do número de símbolos 0's.

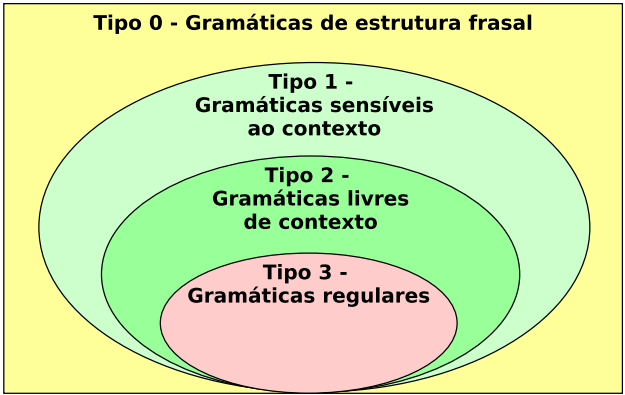
(6) Elaborar uma uma gramática de estrutura frasal que gere todos os palíndromos sobre $\Sigma = \{0, 1\}$.

- **Classificação da gramáticas**
  - **Tipo 0**
    - São as **denominadas gramáticas irrestritas** ou **gramáticas de estrutura frasal**;
    - A única **restrição imposta** por esta **gramática** é que as produções devem ser da **forma** $\alpha \rightarrow \beta$ onde α dever ter **pelo menos um símbolo não terminal**: $\alpha \in V^* \times (V - \Sigma) \times V^*$ e $\beta \in V^*$;
    - Geram linguagens **reconhecidas** por **Máquinas de Turing**.
  - **Tipo 1**
    - São as denominadas **gramáticas sensíveis ao contexto**. As produções deste tipo de gramática devem ser da **forma** $\alpha \rightarrow \beta$ onde $|\alpha| \leq |\beta|$ e $\alpha \in V^* \times (V - \Sigma) \times V^*$ e $\beta \in V^*$;
    - **Outro modo** de descrever este tipo de gramática é **representar** as **produções** do tipo $\alpha \rightarrow \beta$ com $\alpha = \phi_1 A \phi_2$ e $\beta = \phi_1 \psi \phi_2$ (ϕ_1 e ϕ_2 possivelmente vazios) e com ψ não vazio;
    - Assim, A é **reescrito** como ψ no **contexto** de ϕ_1 e ϕ_2, daí o nome “sensível ao contexto” da gramática;
    - Geram linguagens **reconhecidas** por **Autômatos Linearmente Limitados** (tipo de Máquina de Turing não determinística).

Hierarquia das linguagens

- **Classificação da gramáticas**
  - **Tipo 2**
    - São as denominadas **gramáticas livres de contexto**. É o **tipo mais utilizado** na descrição das **linguagens de programação**;
    - As **produções** são da **forma** $\alpha \rightarrow \beta$ onde $\alpha \in V - \Sigma$ e $\beta \in V^*$, o que torna a **gramática livre** de **contexto**;
    - Geram linguagens **reconhecidas** por **Autômatos de Pilha**.
  - **Tipo 3**
    - São as denominadas **gramáticas regulares** ou **lineares**. Geram as mesmas cadeias que **expressões regulares** – uma notação compacta que é empregada no reconhecimento de cadeias em compiladores, interpretadores, mecanismos de busca etc;
    - As produções são da **forma** $\alpha \rightarrow \beta$, $\alpha \in V - \Sigma$ e β possui a forma $\mathtt{a}B$ ou $\mathtt{a}$, onde $\mathtt{a} \in \Sigma$ e $B \in V - \Sigma$;
    - Geram linguagens **reconhecidas** por **Autômatos Finitos Determinísticos**.

- **Hierarquia**


Tipo 0 - Gramáticas de estrutura frasal
Tipo 1 -
Gramáticas sensíveis
ao contexto
Tipo 2 -
Gramáticas livres
de contexto
Tipo 3 -
Gramáticas regulares

Teste seus conhecimentos

(1) Seja $V = \{S, A, B, \mathtt{a}, \mathtt{b}\}$. Determinar se $G = (V, \Sigma, R, S)$ é do tipo 0 (mas não do tipo 1), tipo 1 (mas não do tipo 2), tipo 2 (mas não do tipo 3) ou tipo 3, se R é um conjunto de produções como:

a) $R = \{S \to \mathtt{a}AB, A \to B\mathtt{b}, B \to \epsilon\}$.
b) $R = \{S \to \mathtt{a}A, \mathtt{a}A \to B, B \to \mathtt{a}A, A \to \mathtt{b}\}$
c) $R = \{S \to \mathtt{a}A, A \to \mathtt{b}B, B \to \mathtt{b}, B \to \epsilon\}$

Referências bibliográficas

[1] GERSTING, J. **Fundamentos Matemáticos para a Ciência da Computação: um Tratamento Moderno de Matemática Discreta**. [S.l.]: Livros Técnicos e Científicos. ISBN 9788521614227.

[2] RICH, E. **Automata, Computability and Complexity: Theory and Applications**. [S.l.]: Pearson Prentice Hall, 2008.

[3] ROSEN, K. **Discrete Mathematics and Its Applications**. New York: McGraw-Hill, 2003. (McGraw-Hill higher education).

[4] TAYLOR, R. G. **Models of computation and formal languages**. New York: Oxford University Press, 1998.