Curso de 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 – Escola de Engenharia Mauá
Prof. Marco Antonio Furlan de Souza
<marco.furlan@maua.br>

Agenda

- Modelos de computação
- Gramáticas e linguagens

Algoritmos e computação

- **O conceito de algoritmo**
  - Um algoritmo descreve um **procedimento computacional** que deve ser:
    - **Definido**: em cada ponto em que estiver, sempre há um **próximo passo** determinado – é processo **determinístico**;
    - **Finito**: após um **conjunto finito** de passos, o **algoritmo termina**;
    - Se o algoritmo termina ele deve ser **conclusivo** – gerar um resultado não ambíguo.
  - A questão é: **como modelar um computador** e seu **processo de execução de algoritmos**?
  - Para isso, é necessário **abstrair** o **processo de computação** por si próprio.

Modelo geral de computação

- **Computador como função de transição**
  - **Qualquer computador** (ou sistema físico) pode ser modelado como tendo, em um certo **instante** de **tempo**, um **estado** específico $s \in S$ de algum **espaço de estados** S (finito or infinito);
  - E, em **qualquer instante**, o computador pode **receber** um **símbolo** de **entrada** $i \in I$ e **produzir** um **símbolo** de **saída** $o \in O$, onde I e O são conjuntos de símbolos (cada símbolo pode codificar uma quantidade arbitrária de dados);
  - Um **computador** pode, então, ser **modelado** simplesmente como uma **função de transição** $T : S \times I \rightarrow S \times O$, onde I e O são conjuntos de símbolos;
  - Todo modelo de computação pode ser visto como um caso especial desta formulação geral.

Modelos de computação

- **Três modelos** importantes:
  - **Gramáticas e linguagens**: utilizadas para gerar palavras de uma **linguagem** e para **reconhecer** se uma **palavra** faz parte da **linguagem**. Gramáticas são muito importantes na **teoria** e **construção** de **compiladores**;
  - **Máquinas de estado finito**: possuem diversos tipos e são caracterizadas por um conjunto de **estados**, um **alfabeto** de entrada e uma **função** de **transição de estados** que mapeia um **próximo estado** a um par composto por **estado** e **entrada** atuais. São utilizadas para **reconhecimento de linguagens** e modelagem de diversos tipos de máquinas;
  - **Máquinas de Turing**: são utilizadas para reconhecer **conjuntos de expressões** complexos que não podem ser reconhecidos por máquinas de estados finito e também para **computar funções** numéricas teóricas.

Agenda

- Modelos de computação
- Gramáticas e linguagens

Alfabetos e cadeias

- **Alfabeto**

Definição. *Um* ***alfabeto*** *ou* ***vocabulário****, escrito como Σ, é um* ***conjunto finito*** *e* ***não vazio*** *de* ***símbolos****.*

- **Cadeia**

Definição. *Uma* ***cadeia*** *é uma* ***sequência****, possivelmente vazia, de* ***comprimento finito*** *de elementos de Σ. A* ***cadeia vazia*** *ou* ***cadeia nula****, denotada por ϵ, é uma cadeia que não contém símbolo algum.*

O ***conjunto de todas as cadeias*** *sobre Σ é denotado por Σ^*. A cadeia vazia sempre pertence à Σ^* e, se não for o caso, utilizar a notação Σ^+, que representa $\Sigma^* - \{\epsilon\}$.*

Uma ***linguagem*** *sobre Σ é um subconjunto de Σ^*, $L \subseteq \Sigma^*$.*

Alfabetos e cadeias

- **Exemplos**
  - Alfabeto binário: $\Sigma = \{\mathtt{0}, \mathtt{1}\}$;
  - Letras minúsculas do alfabeto: $\Sigma = \{\mathtt{a}, \mathtt{b}, \dots, \mathtt{z}\}$;
  - Se $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, $\Sigma^0 = \{\epsilon\}$, $\Sigma^1 = \{\mathtt{0}, \mathtt{1}\} = \Sigma$, $\Sigma^2 = \{\mathtt{00}, \mathtt{01}, \mathtt{10}, \mathtt{11}\}$, $\Sigma^3 = \{\mathtt{000}, \mathtt{001}, \mathtt{010}, \mathtt{011}, \mathtt{100}, \mathtt{101}, \mathtt{110}, \mathtt{111}\}$;
  - O conjunto de todas as palavras sobre um alfabeto Σ é definido por $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$;
  - Se $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, $\Sigma^* = \{\epsilon, \mathtt{0}, \mathtt{1}, \mathtt{00}, \mathtt{01}, \mathtt{10}, \mathtt{11}, \mathtt{000}, \dots\}$;
  - Se $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, $\Sigma^+ = \{\mathtt{0}, \mathtt{1}, \mathtt{00}, \mathtt{01}, \mathtt{10}, \mathtt{11}, \mathtt{000}, \dots\}$.

Funções sobre cadeias

- **Comprimento**

  Definição. *O **comprimento** de uma cadeia s é escrita como $|s|$ é o número de símbolos presentes em s. O comprimento da cadeia vazia é zero, $|\epsilon| = 0$;*

- **Concatenação**

  Definição. *A **concatenação** de duas cadeias s e t é escrita como st (ou $s||t$) e resulta em uma **cadeia** formada pela **justaposição** de t à s. O comprimento de uma cadeia obtida pela concatenação de duas cadeias s e t é igual à soma dos comprimentos de s e t: $|st| = |s| + |t|$.*

Funções sobre cadeias

- **Replicação**

  Definição. *Para cada **cadeia** w e para cada **número natural** i, a cadeia w^i é definida como:*

$$w^0 = \epsilon$$
$$w^{i+1} = w^i w$$

- **Inversa de uma cadeia**

  Definição. *Para cada cadeia w, a cadeia **inversa** de w, escrita como w^R é assim definida:*

$$w^R = \begin{cases} \epsilon & \textit{se } |w| = 0 \\ au^R & \textit{se } |w| \geq 1 \land (\exists a)(\exists u)(a \in \Sigma \land u \in \Sigma^* \land w = ua) \end{cases}$$

Funções sobre cadeias

- **Exemplos**
  - $|\epsilon| = 0$;
  - $|\texttt{1001101}| = 7$;
  - Se $x = $good e $y = $bye então $xy = $goodbye;
  - $\mathsf{a}^0\mathsf{b}^3 = \mathsf{bbb}$;
  - $\mathsf{a}^3 = \mathsf{aaa}$;
  - $(\mathsf{bye})^2 = \mathsf{byebye}$;
  - Se $s = \texttt{roma}$, então

$$
\begin{aligned}
\texttt{roma}^R &= \texttt{a}(\texttt{rom})^R \\
&= \texttt{am}(\texttt{ro})^R \\
&= \texttt{amo}(\texttt{r})^R \\
&= \texttt{amor}(\epsilon)^R \\
&= \texttt{amor}
\end{aligned}
$$

Relações sobre cadeias

- **Subcadeia e subcadeia própria**

  Definição. *Uma cadeia s é uma* ***subcadeia*** *de t se e somente se s* ***ocorre contiguamente*** *como parte de t.*

  Uma cadeia s é uma ***subcadeia própria*** *de t se e somente se s ocorre contiguamente como parte de t e $s \neq t$.*

- **Prefixo e prefixo próprio**

  Definição. *Uma cadeia s é um* ***prefixo*** *de t se e somente se $(\exists x \in \Sigma^*)(t = sx)$.*

  Uma cadeia s é um ***prefixo próprio*** *de t se e somente se s é prefixo de t e $s \neq t$.*

Relações sobre cadeias

- **Sufixo e sufixo próprio**

Definição. *Uma cadeia s é um **sufixo** de t se e somente se* $(\exists x \in \Sigma^*)(t = xs)$.

*Uma cadeia s é um **sufixo próprio** de t se e somente se s é sufixo de t e e $s \neq t$.*

Relações sobre cadeias

- **Exemplos**
  - aaa é subcadeia (própria) de bbaaaacc;
  - bbaaaacc é subcadeia de bbaaaacc;
  - bbaaaacc não é subcadeia própria de bbaaaacc;
  - bba é prefixo (próprio) de bbaaaacc;
  - acc é sufixo (próprio) de bbaaaacc.

Linguagens

- **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 ou
    - **Um reconhecedor da linguagem**: que decida se uma cadeia está ou não na linguagem, retornando `V` se estiver e `F`, caso contrário.
  - Este problema, embora pareça simples, é tão geral quanto a própria noção de computação.

Linguagens

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

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

  - Cadeias que terminam em a:

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

  - Linguagem vazia (não contém cadeia alguma):

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

  - Linguagem contendo apenas a cadeia vazia:

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

Linguagens

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

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

  - Nenhum prefixo começa com b:

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

  - Todo prefixo começa com b:

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

Linguagens

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

$$\begin{aligned} L &= \{\mathsf{a}^n : n \geq 0\} \\ &= \{\epsilon, \mathsf{a}, \mathsf{aa}, \mathsf{aaa}, \mathsf{aaaa}, \ldots\} \end{aligned}$$

  - Exemplo já apresentado, reescrito:

$$\begin{aligned} L &= \{\mathsf{a}^m \mathsf{b}^n : m, n \geq 0\} \\ &= \{\epsilon, \mathsf{a}, \mathsf{aa}, \mathsf{aabbb}, \mathsf{bbb}, \ldots\} \end{aligned}$$

Operações sobre linguagens

- **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. $\square$

Operações sobre linguagens

- **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**. $\square$

Operações sobre linguagens

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

$$\Sigma = \{\mathsf{a}, \mathsf{b}\}$$
$$L_1 = \{\text{cadeias com um número par de } \mathsf{a}\text{'s}\}$$
$$L_2 = \{\text{cadeias sem } \mathsf{b}\text{'s}\}$$

Então:

$$L_1 \cup L_2 = \{\text{todas as cadeias com apenas } \mathsf{a}\text{'s e cadeias contendo } \mathsf{b}\text{'s com um número par de } \mathsf{a}\text{'s}\}$$
$$L_1 \cap L_2 = \{\epsilon, \mathsf{aa}, \mathsf{aaaa}, \mathsf{aaaaaa}, \ldots\}$$
$$L_2 - L_1 = \{\mathsf{a}, \mathsf{aaa}, \mathsf{aaaaa}, \ldots\}$$
$$\neg(L_2 - L_1) = \{\text{cadeias com no mínimo um } \mathsf{b}\} \cup \{\text{cadeias com número par de } \mathsf{a}\text{'s}\}$$

Operações sobre linguagens

- **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:

$$L_1 = \{\texttt{cat}, \texttt{dog}, \texttt{mouse}, \texttt{bird}\}$$

$$L_2 = \{\texttt{bone}, \texttt{food}\}$$

$$L_1L_2 = \{\texttt{catbone}, \texttt{catfood}, \texttt{dogbone}, \texttt{dogfood}, \texttt{mousebone}, \texttt{mousefood}, \texttt{birdbone}, \texttt{birdfood}\}$$

Operações sobre linguagens

- **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**:

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

  - 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: $L^+ = LL^*$.

Operações sobre linguagens

- **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.

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

□

Gramáticas

- **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 popular de descrever gramáticas recursivamente é a **gramática de estrutura frasal**, elaborada por Noam Chomsky na década de 1950.

Gramáticas

- **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* $\alpha \to \beta$*;*
- S *é o símbolo não-terminal **de partida** ou **inicial**,* $S \in V - \Sigma$*.*

Gramáticas

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

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

Derivação

- **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* $\psi \stackrel{*}{\Rightarrow} \sigma$ *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* $\stackrel{*}{\Rightarrow}$ *é o fecho transitivo reflexivo da relação* $\Rightarrow$*.*

Derivação

- **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:

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

  - Então, $S \overset{*}{\Rightarrow} \mathtt{a13}$.
  - 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 $L = (\mathtt{11})^*\mathtt{0}$, o conjunto de todas as cadeias consistindo de algum número de concatenações de $\mathtt{11}$ com ele próprio, seguido finalmente por $\mathtt{0}$.

(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 $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

Hierarquia das linguagens

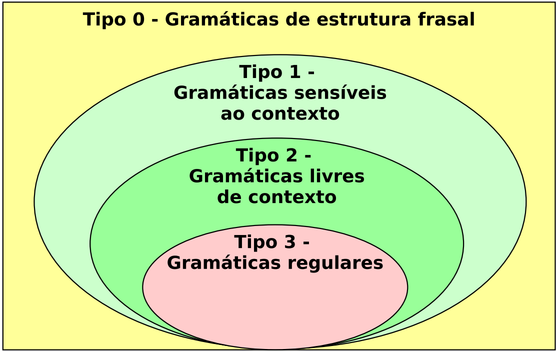
- **Classificação da gramáticas**
  - **Tipo 0**. São as denominadas **gramáticas irrestritas** ou **gramáticas de estrutura frasal**. A única restrição imposta por esta gramática é que as produções devem ter pelo menos um símbolo não-terminal em sua composição.
  - **Tipo 1**. São as denominadas **gramáticas sensíveis ao contexto**. As produções deste tipo de gramática devem ser da forma $\alpha \rightarrow \beta$ onde $|\alpha| < |\beta|$ (evita que β seja vazia). Outra forma 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.

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 $\alpha \to \beta$ onde $|\alpha| \leq |\beta|$ e $\alpha \in V - \Sigma$, o que torna a gramática livre de contexto.
  - **Tipo 3** . São as denominadas **gramáticas regulares** ou **lineares**. É muito utilizada **expressões regulares**, uma notação compacta que é empregada no reconhecimento de cadeias em compiladores, interpretadores, mecanismos de busca etc. As produções são da forma $\alpha \to \beta$ onde $|\alpha| \leq |\beta|$, $\alpha \in V - \Sigma$ e β possui a forma aB ou a, onde a $\in \Sigma$ e $B \in V - \Sigma$.

Hierarquia das linguagens

- **Hierarquia**



Teste seus conhecimentos

(1) Seja $V = \{S, A, B, \mathsf{a}, \mathsf{b}\}$. 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) $R = \{S \to \mathsf{a}AB, A \to B\mathsf{b}, B \to \epsilon\}$.

b) $R = \{S \to \mathsf{a}A, \mathsf{a}A \to B, B \to \mathsf{a}A, A \to \mathsf{b}\}$

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

Referências bibliográficas

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

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

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

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