Linguagens Formais e Autômatos

Notas de Aula

Valdigleis S. Costa
Universidade Federal do Rio Grande do Norte – UFRN
Centro de Ciências Exatas e da Terra – CCET
Departamento de Informática e Matemática Aplicada – DIMAP

12 de agosto de 2025

Este tomo foi escrito com base em uma coleção de notas de aulas do autor, o mesmo foi redigido usando um *template* desenvolvido pelo próprio autor. Este texto foi escrito com o conjunto de macros LaTeX (em sua versão 2) e compilado usado as ferramentas LuaLaTeX e BibTeX, tais ferramentas fornecidas pelas distribuições TeXLive e MacTeX, respectivamente nos sistema operacionais *Unix-like*: Debian e no Mac OS X, para edição foram usados os *softwares* livres de edição textual Vim (versão 0.10.1), além disso, o sistema de controle de versão adotado é o Git (versão 2.34.1).

Release compilado em 12 de agosto de 2025 (1160 minutos após a meia-noite).

Sumário

Parte I

Fundamentos

Capítulo 1

Sobre Autômatos e Linguagens

"Ciência é uma equação diferencial.
Religião é a condição de contorno."

Alan M. Turing.

1.1 Introdução

O conceito de linguagem formal é estabelecido como sendo um conjunto (possivelmente infinito) de palavras (ou sentenças) definidas sobre um dado alfabeto. Cada palavra em uma linguagem formal é simplesmente uma sequência finita de símbolos presente no alfabeto alfabeto em questão. As palavras de uma linguagem formal, em alguns cenários são também pode ser chamadas de termos, ou ainda, de fórmulas [18].

A sintaxe de qualquer linguagem formal é especificada por um conjunto finito de regras bem definidas, tal conjunto recebe o nome de gramática. A gramática é de fato o mecanismo que determinam a estrutura das palavras que podem existir dentro da linguagem, o que faz com que não exista nenhum grau de liberdade na forma (o por isso a nomenclatura **formal**) das palavras[5], ou seja, todas as palavras devem seguir uma forma rigorosa.

No que diz respeito a palavras em uma linguagem formal, elas (como já foi dito) são apenas sequência de símbolos sem qualquer significado. Para atribuir um significado as palavras de uma linguagem formal, isto é, para atribuir semântica a linguagem formal, é obrigatoriamente necessário definir externamente[1] uma estrutura avaliativa a semântica da linguagem, e de fato, isso é exatamente o que ocorrer durante a construção de compiladores e interpretadores para as linguagens de programação (detalhes em [2, 9]), essa estrutura externa funciona como um universo avaliativo, no qual as palavras da linguagem pode ser interpretadas (ter seu significado exposto).

[1] No sentido de que a interpretação das palavras é feita fora de sua gramática geradora.

Neste documento o foco será descrever uma teoria para computabilidade nos padrões apresentados por Turing [25], ou seja, uma visão de computabilidade por máquinas de computação, os chamados autômatos finitos [2, 13]. Além desse objetivo, essa parte do documento também irá forma uma pequena teoria para as linguagens formais usando modelos de representação, computação e geração, para a diferentes classes de linguagens. O que é importante para o leitor interessado em aprender sobre construção de compiladores e linguagens de programação, assim neste primeiro capítulo serão apresentados alguns conceitos básicos necessários nos capítulos seguintes.

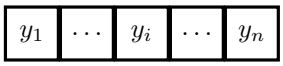
1.2 Sobre Autômatos Finitos

Como dito em [10, 11] uma definição informal do conceito de autômato finito (ou máquina de estado finita) e que tais dispositivos podem ser vistos como sendo máquinas com dois componentes fundamentais:

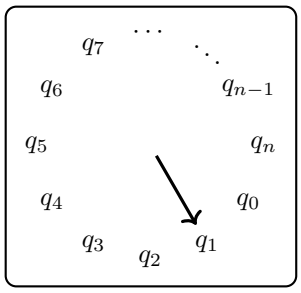
- Um conjunto finito de memórias[2], estas sendo subdivididas em células, cada uma das quais capaz de comportar um único símbolo por vez.
- Uma unidade de controle[3] que administra o estado atual do autômato e é responsável por executar as instruções (programa) da máquina.

[2] Na literatura também é usado o termo fita em vez de memória [21].

[3] Também chamada de unidade de processamento (UCP).



Memória do autômato



Unidade de Controle

Figura 1.1: Representação informal do conceito de autômato finito com uma única memória retirado de [11].

Com respeito as memórias é comum assumir a existência de um **dispositivo de leitura e (ou) escrita**[1] que é capaz de acessar uma única célula por vez, e assim pode lê e (ou) escrever na célula. A depender do tipo de autômato podem existir vários dispositivos de leitura/escrita ou apenas um [6].

A(s) memória(s) de um autômato finito serve(m) para guarda dados (os símbolos) usados durante o funcionamento do autômato. O funcionamento de um autômato por sua vez, pode ser descrito em tempo discreto [10, 11], assim sendo, em qualquer momento no tempo t, a **unidade de controle** do autômato estará sempre em algum **estado** interno possível e a(s) **unidade(s) de leitura/escrita** tem acesso a alguma(s) **célula(s)** da(s) memória(s).

Formalmente pode-se dizer como apontado em [11], que a teoria dos autômatos finitos, ou simplesmente teoria dos autômatos, teve seu desenvolvimento inicial entre os anos de 1940 e 1960 sendo este início os trabalhos de McCulloch e Pitts [19], Kleene [14], Mealy [20], Moore [22], Rabin e Scott [23, 24].

1.3 Noções Fundamentais

Neste primeiro momento para o estudo dos autômatos finitos serão apresentados alguns conceitos fundamentais de extrema importância para o desenvolvimento das próximas seções e capítulos.

Definição 1 (Alfabetos e Palavras) [10] Qualquer conjunto finito e não vazio Σ será chamado de alfabeto. Qualquer sequência finita de símbolos na forma $a_1 \cdots a_n$ com $a_i \in \Sigma$ para todo $1 \leq i \leq n$ será chamada de palavra sobre o alfabeto Σ.

[1] Também é usado a nomenclatura cabeçote [10, 11].

Exemplo 1 | Os conjuntos $\{0,1,2,3\}, \{a,b,c\}, \{\heartsuit, \spadesuit, \diamond, \clubsuit\}$ e $\{n \in \mathbb{N} \mid n \leq 25\}$ são todos alfabetos, os conjuntos $\mathbb{N}$ e $\mathbb{R}$ não são alfabetos.

Exemplo 2 | Dado o alfabeto $\Sigma = \{0,1,2,3\}$ tem-se que as sequências 0123, 102345, 1 e 0000 são todas palavras sobre Σ.

Definição 2 (Comprimento das palavras) Seja w uma palavra qualquer sobre um certo alfabeto Σ, o comprimento[a] de w, denotado por $|w|$, corresponde ao número de símbolos existentes em w.

[a] Por conta desta notação em alguns texto é usado o termo módulo em vez de comprimento.

Exemplo 3 | Dado o alfabeto $\Sigma = \{a,b,c,d\}$ e as palavras $abcd, aacbd, c$ e $ddaacc$ tem-se que: $|abcd| = 4, |aa| = 2, |c| = 1$ e $|ddaacc| = 6$.

Como muito bem explicado em [6, 13, 15], pode-se definir uma série de operações sobre palavras, sendo a primeira delas a noção de concatenação.

Definição 3 (Concatenação de palavras) Sejam $w_1 = a_1 \cdots a_m$ e $w_2 = b_1 \cdots b_n$ duas palavras quaisquer, tem-se que a concatenação de w_1 e w_2, denotado por $w_1 w_2$, corresponde a uma sequência iniciada com os símbolos que forma w_1 imediatamente seguido dos símbolos que forma w_2, ou seja, $w_1 w_2 = a_1 \cdots a_m b_1 \cdots b_n$.

É importante notar que a concatenação apenas combina duas palavras em uma nova palavra, sendo que, não existe qualquer tipo de exigência sobre os alfabeto sobre os quais as palavras usadas na concatenação estão definidas, ou seja, **podem ser alfabetos distintos**.

Exemplo 4 | Dado duas palavras $w_1 = abra$ e $w_2 = cadabra$ tem-se que $w_1 w_2 = abracadabra$ e $w_2 w_1 = cadabraabra$.

Note que o Exemplo 4 estabelece que a operação de concatenação entre duas palavras não é comutativa, isto é, a ordem com que as palavras aparecem na concatenação é responsável pela forma da palavra resultante da concatenação.

Teorema 1 (Associativade da Concatenação) Para quaisquer w_1, w_2 e w_3 tem-se que $(w_1 w_2)w_3 = w_1(w_2 w_3)$.

Prova | Dado três palavras quaisquer $w_1 = a_1 \cdots a_i, w_2 = b_1 \cdots b_j$ e $w_3 = c_1 \cdots c_k$ tem-se que,

$$\begin{aligned}(w_1 w_2)w_3 &= (a_1 \cdots a_i b_1 \cdots b_j)c_1 \cdots c_k \\ &= a_1 \cdots a_i b_1 \cdots b_j c_1 \cdots c_k \\ &= a_1 \cdots a_i (b_1 \cdots b_j c_1 \cdots c_k) \\ &= w_1(w_2 w_3)\end{aligned}$$

o que conclui a prova. □

Sobre qualquer alfabeto Σ sempre é definida uma palavra especial chamada **palavra vazia** [13, 15], essa palavra especial não possui nenhum símbolo, e em geral é usado o símbolo λ para denotar a palavra vazia [6, 10]. Como mencionado em [6, 11] sobre a palavra vazia é importante destacar que:

$$w\lambda = \lambda w = w \tag{1.1}$$
$$|\lambda| = 0 \tag{1.2}$$

Isto é, a palavra vazia é neutra para a operação de concatenação, além disso, a mesma apresenta comprimento nulo.

Definição 4 (Potência das palavras) Seja w uma palavra sobre um alfabeto Σ a potência de w é definida recursivamente para todo $n \in \mathbb{N}$ como sendo:

$$w^0 = \lambda \tag{1.3}$$
$$w^{n+1} = ww^n \tag{1.4}$$

Exemplo 5 Sejam $w_1 = ab, w_2 = bac$ e $w_3 = cbb$ palavras sobre $\Sigma = \{a, b, c\}$ tem-se que:

(a) $w_1^3 = w_1w_1^2 = w_1w_1w_1^1 = w_1w_1w_1w_1^0 = w_1w_1w_1\lambda = ababab.$

(b) $w_2^2 = w_2w_2^1 = w_2w_2w_2^0 = w_2w_2\lambda = w_2w_2 = bacbac.$

Exemplo 6 Seja $u = 01$ e $v = 231$ tem-se que:

$$uv^3 = uvv^2 = uvvv^1 = uvvv\lambda = uvvv = 01231231231$$

e também

$$u^2v = uu^1v = uu\lambda v = uuv = 0101231$$

Lema 1 Para toda palavra w e todo $m, n \in \mathbb{N}$ tem-se que:

(i) $(w^m)^n = w^{mn}$.

(ii) $w^mw^n = w^{m+n}$.

Prova Direto das Definições 3 e 4, e portanto, ficará como exerício ao leitor. □

Outro importante conceito existente sobre a ideia de palavra é a noção de palavra inversa (ou reversa) formalmente definida como se segue.

Definição 5 (Palavra Inversa) [10] Seja $w = a_1 \cdots a_n$ uma palavra qualquer, a palavra inversa de w denotada por w^r, é tal que $w^r = a_n \cdots a_1$.

Exemplo 7 Dado as palavras $u = aba, v = 011101$ e $w = 3021$ tem-se que $u^r = aba, v^r = 101110$ e $w^r = 1203$.

Além das palavras, pode-se também formalizar uma série de operações sobre a própria noção de alfabeto. Em primeiro lugar, uma vez que, alfabetos são conjuntos, obviamente todas operações usuais de união, interseção, complemento, diferença e diferença simétrica também são válidas sobre alfabetos. Além dessas operações, também esta definida a operação de potência e os fechos positivo e de Kleene sobre alfabetos.

Definição 6 (Potência de um alfabeto) [6] Seja Σ um alfabeto a potência de Σ é definida recursivamente para todo $n \in \mathbb{N}$ como:

$$\Sigma^0 = \{\lambda\} \tag{1.5}$$
$$\Sigma^{n+1} = \{aw \mid a \in \Sigma, w \in \Sigma^n\} \tag{1.6}$$

Exemplo 8 Dado $\Sigma = \{a, b\}$ tem-se que $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$ e $\Sigma^1 = \{a, b\}$

Exemplo 9 Seja $\Sigma = \{0, 1, 2\}$ tem-se que $\Sigma^2 = \{00, 01, 02, 10, 11, 12, 20, 21, 22\}$ e $\Sigma^0 = \{\lambda\}$.

O leitor mais atencioso e maduro matematicamente pode notar que para qualquer que seja $n \in \mathbb{N}$ o conjunto potência tem a propriedade de que todo $w \in \Sigma^n$ é tal que $|w| = n$, além disso, é claro que todo Σ^n é sempre finito, uma vez que, o conjunto Σ^n pode ser visto como sendo nada mais do que, um conjunto de arranjos com repetição.

Definição 7 (Fecho Positivo e de Kleene) Seja Σ um alfabeto o fecho positivo e o fecho de Kleene de Σ, denotados respectivamente por Σ^+ e Σ^*, correspondem aos conjuntos: $\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i$ e $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$.

Obviamente como dito em [6], o fecho positivo pode ser reescrito em função do fecho de Kleene usando a operação de diferença de conjunto, isto é, o fecho positivo corresponde a seguinte identidade, $\Sigma^+ = \Sigma^* - \{\lambda\}$. Sobre o fecho de Kleene como destacado em [11] o mesmo corresponde ao monoide livremente gerado pelo conjunto Σ munida da operação de concatenação.

Definição 8 (Prefixos e Sufixos) Uma palavra $u \in \Sigma^*$ é um prefixo de outra palavra $w \in \Sigma^*$, denotado por $u \preceq_p w$, sempre que $w = uv$, com $v \in \Sigma^*$. Por outro lado, uma palavra u é um sufixo de outra palavra w, denotado por $u \preceq_s w$, sempre que $w = vu$.

Exemplo 10 Seja $w = abracadabra$ tem-se qu e as palavras ab e $abrac$ são prefixos de w, por outro, lado $cadabra$ e bra são sufixos de w, e a palavra $abra$ é prefixo e também sufixo. Já a palavra $cada$ não é prefixo e nem sufixo de w.

Definição 9 (Conjunto dos Prefixos e Sufixos) Seja $w \in \Sigma^*$ o conjunto de todos os prefixos de w corresponde ao conjunto:

$$PRE(w) = \{w' \in \Sigma^* \mid w' \preceq_p w\} \tag{1.7}$$

e o conjunto de todos os sufixos de w corresponde ao conjunto:

$$SUF(w) = \{w' \in \Sigma^* \mid w' \preceq_s w\} \tag{1.8}$$

Exemplo 11 Seja $w = univasf$ tem-se que:

$$PRE(w) = \{\lambda, u, un, uni, univ, univa, univas, univasf\}$$

e

$$SUF(w) = \{\lambda, f, sf, asf, vasf, ivasf, nivasf, univasf\}$$

Exemplo 12 A seguir é apresentado alguns exemplos de palavras e seus conjuntos de prefixos e sufixos.

(a) Se $w = ab$, então $PRE(w) = \{\lambda, a, ab\}$ e $SUF(w) = \{\lambda, b, ab\}$.

(b) Se $w = 001$, então $PRE(w) = \{\lambda, 0, 00, 001\}$ e $SUF(w) = \{\lambda, 1, 01, 001\}$.

(c) Se $w = \lambda$, então $PRE(w) = \{\lambda\}$ e $SUF(w) = \{\lambda\}$

(d) Se $w = a$, então $PRE(w) = \{\lambda, a\}$ e $SUF(w) = \{\lambda, a\}$.

Com respeito a cardinalidade dos conjuntos de prefixos e sufixos, os mesmo apresentam as propriedades descritas pelo teorema a seguir.

Teorema 2 Para qualquer que seja $w \in \Sigma^*$ as seguintes asserções são verdadeiras.

(i) $\#PRE(w) = |w| + 1$.

(ii) $\#PRE(w) = \#SUF(w)$.

(iii) $\#(PRE(w) \cap SUF(w)) > 1$.

Prova Dado uma palavra w tem-se que:

(i) Sem perda de generalidade assumindo que $w = a_1 \cdots a_n$ logo $w \in \Sigma^n$ (o caso quando $w = \lambda$ é trivial e não será demonstrado aqui) logo $|w| = n$ para algum $n \in \mathbb{N}$, assim existem exatamente n palavras da forma $a_1 \cdots a_i$ com $1 \leq i \leq n$ tal que $a_1 \cdots a_i \preceq_p w$, portanto, para todo $1 \leq i \leq n$ tem-se que $a_1 \cdots a_i \in PRE(w)$, além disso, é claro que $w = \lambda w$, e portanto, $\lambda \in PRE(w)$, consequentemente, $\#PRE(w) = n + 1 = |w| + 1$.

(ii) Analoga ao item anterior.

(iii) Trivial, pois basta notar que $\lambda, w \in (PRE(w) \cap SUF(w))$, e portanto, tem-se claramente que $\#(PRE(w) \cap SUF(w)) > 1$. □

Corolario 1 Toda palavra tem pelo menos um prefixo e um sufixo.

Prova Direto do item (iii) do Teorema 2. □

Seguindo com este documento pode-se finalmente formalizar o pilar fundamental (a ideia de linguagem) necessário para desenvolver o estuda da computabilidade neste e nos próximo capítulos.

Definição 10 (Linguagem) Dado um alfabeto Σ, qualquer subconjunto $L \subseteq \Sigma^*$ será chamado de linguagem.

Exemplo 13 Seja $\Sigma = \{0, 1\}$ tem-se que os conjuntos a seguir são todos linguagens sobre Σ.

(a) Σ^*.

(b) $\{0^n b^n \mid n \in \mathbb{N}\}$.

(c) $\{\lambda, 0, 1\}$.

(d) Σ^{22}.

(e) $\emptyset$.

Similarmente ao que ocorre com os alfabetos, as linguagens por serem conjuntos "herdam" as operações básicas da teoria dos conjuntos [16, 17, 1], isto é, estão definidas sobre as linguagens as operações de união, interseção, completo, diferença e diferença simétrica. E como par aos alfabetos novas operações são definidas.

Definição 11 (Concatenação de Linguagens) Sejam L_1 e L_2 duas linguagens, a concatenação de L_1 com L_2, denotado por $L_1 L_2$, corresponde a seguinte linguagem:

$$L_1 L_2 = \{xy \in (\Sigma_1 \cup \Sigma_2)^* \mid x \in L_1, y \in L_2\} \tag{1.9}$$

Exemplo 14 Dado as três linguagens $L_1 = \{\lambda, ab, bba\}, L_2 = \{0^{2n}1 \mid n \in \mathbb{N}\}$ e $L_3 = \{a^p \mid p \text{ é um número primo}\}$ tem-se que:

(a) $L_1 L_2 = \{w \mid w = 0^{2n}1 \text{ ou } w = ab0^{2n}1 \text{ ou } w = bba0^{2n}1 \text{ com } n \in \mathbb{N}\}$.

(b) $L_3 L1 = \{w \mid w = a^p \text{ ou } w = a^{p+1}b \text{ ou } a^p bba \text{ onde } p \text{ é um número primo}\}$.

(c) $L_2 L_3 = \{0^{2n}1a^p \mid n \in \mathbb{N}, p \text{ é um número primo}\}$.

Definição 12 (Linguagem Reversa) Seja L uma linguagem, a linguagem inversa de L, denotada por L^r, corresponde ao conjunto $\{w^r \mid w \in L\}$.

Exemplo 15 Considerando as linguagens L_1, L_2 e L_3 do Exemplo 14 tem-se que:

(a) $L_1^r = \{\lambda, ba, abb\}$.

(b) $L_2^r = \{10^{2n} \mid n \in \mathbb{N}\}$.

(c) $L_3^r = \{a^p \mid n \in \mathbb{N}, p \text{ é um número primo}\}$.

O leitor mais atento pode perceber que a propriedade involutiva da operação reversa sobre palavras é "herdada" para a reversão sobre linguagens, isto é, para qualquer linguagem L tem-se que $(L^r)^r = L$.

Definição 13 (Linguagem Potência) Seja L uma linguagem, a linguagem potência de L, denotada por L^n, é definida recursivamente para todo $n \in \mathbb{N}$ como:

$$L^0 = \{\lambda\} \tag{1.10}$$
$$L^{n+1} = LL^n \tag{1.11}$$

Utilizando o conceito de linguagem potência a seguir é apresentado a formalização para os fechos positivo e de Kleene sobre linguagens.

Definição 14 (Fecho positivo e Fecho de Kleene de Linguagens) Seja L uma linguagem, o fecho positivo (L^+) e o fecho de Kleene (L^*) de L são dados pelas equações a seguir.

$$L^+ = \bigcup_{i=1}^{\infty} L^i \tag{1.12}$$
$$L^* = \bigcup_{i=0}^{\infty} L^i \tag{1.13}$$

Por fim, esta seção irá apresentar a noção de linguagem dos prefixos e sufixos.

Definição 15 (Linguagem de Prefixos e Sufixos) Seja L uma linguagem, a linguagem dos prefixos e dos sufixos de L, respectivamente $PRE(L)$ e $SUF(L)$, são exatamente os seguintes conjuntos:

$$PRE(L) = \{w' \in \Sigma^* \mid w' \preceq_p w, w \in L\}$$
$$SUF(L) = \{w' \in \Sigma^* \mid w' \preceq_s w, w \in L\}$$

Nos próximos capítulos deste documento irão ser apresentadas as formalizações da ideia de linguagens formais na visão "mecânica" de Turing [25]. Entretanto, em vez de apresentar de forma direta os conceitos ligados as máquinas de Turing e as computações por elas realizadas, este documento opta por fazer um estudo seguindo a ideia dos livros texto de linguagens formais [6, 15, 21], assim sendo, aqui serão apresentadas as linguagens formais da mais simples para a mais complexas seguindo a hierarquia de Chomsky [8], ou seja, serão aqui estudadas as linguagens formais na seguintes ordem: regulares, livres do contexto, recursivas e recursivamente enumeráveis.

1.4 Sobre Gramática Formais

Agora que foram introduzidos os conceitos fundamentais para a teoria das linguagens formais pode-se formalizar o conceito de estrutura geradora ou gramática formal, o leitor mais atento e com maior conhecimento sobre lógica de primeira ordem e teoria da prova [3, 7] pode notar que gramáticas formais são na verdade outro nome para sistemas de reescrita [4].

Definição 16 (Gramática formal) Uma gramática formal é uma estrutura da forma $G = \langle V, \Sigma, S, P\rangle$ onde V é um conjunto não vazio de símbolos chamados variáveis tal que $V \cap \Sigma = \emptyset$, Σ é um alfabeto, $S \in V$ é uma variável destacada chamada de **variável inicial** e P é um conjunto de regras de reescrita[a] da forma $w \to w'$ onde

$w \in (V \cup \Sigma)^+$ e $w' \in (V \cup \Sigma)^*$.

[a] Também é comum encontrar na literatura a nomenclatura regras de produção [6, 15].


Atenção

Na escrita do conjunto P sempre que $w \to w_1$ e $w \to w_2$ com $w_1 \neq w_2$, é escrito simplesmente $w \to w_1 \mid w_2$, em vez de escrever as duas regras separadas.

Exemplo 16 A estrutura $G = \langle \{A, B\}, \{a\}, A, P\rangle$ em que P é formado pelas regras $A \to aABa \mid B$ e $B \to \lambda$ é uma gramática formal.

Qualquer gramática então pode ser visto com um sistema para a geração de palavras através de um mecanismo chamado derivação descrito a seguir.

Definição 17 (Derivação de palavras) Dado uma gramática $G = \langle V, \Sigma, S, P\rangle$, a palavra XwY deriva a palavra $Xw'Y$ na gramática G, denotado por $XwY \vdash_G Xw'Y$, sempre que existe uma regra forma $w \to w' \in P$.

Exemplo 17 Dado a gramática do Exemplo 16 tem-se que $aABa \vdash_G aaABaBa$, pois existe em P a regra $A \to aABa$.

Rigorosamente $\vdash_G$ na verdade é uma relação entre $(V \cup \Sigma)^+$ e $(V \cup \Sigma)^*$, e assim $\vdash_G^*$ denota o fecho transitivo e reflexivo de $\vdash_G$, além disso, sempre que não causar confusão é comum eliminar a escrita do rótulo da gramática, ou seja, são escritos respectivamente $\vdash^*$ e $\vdash$ em vez de $\vdash_G^*$ e $\vdash_G$.

Exemplo 18 Considerando ainda a gramática exibida no Exemplo 16 tem-se que $aABa \vdash^* aaaABaBaBa$, uma vez que, $aABa \vdash aaABaBa \vdash aaaABaBaBa$.

Exemplo 19 A estrutura $G = \langle \{A, B, S\}, \{0, 1\}, S, P\rangle$ em que P é formado pelas regras $S \to 11A, A \to B0$ e $B \to 000$ é uma gramática formal e assim $11A \vdash^* 110000$, pois tem-se que, $11A \vdash^* 11B0 \vdash^* 110000$.

Como dito em [6], dado uma gramática formal G sempre que houver uma sequencia de derivações $w_1 \vdash w_2 \vdash \cdots \vdash w_n$ acontecer, as palavras $w_1, w_2, \cdots, w_n$ são chamadas de formas sentenciais, ou simplesmente, sentenças. Assim uma derivação nada mais é do que uma sequência finita de formas sentenciais.

Definição 18 (Igualdade de Derivações) Dado uma gramática $G = \langle V, \Sigma, S, P\rangle$ e duas derivações $S \vdash w_1 \vdash^* w_n$ e $S \vdash w_1' \vdash^* w_n'$ sobre G, será dito que estas derivações são iguais sempre que $w_i = w_i'$ para todo $1 \leq i \leq n$.

Desde que $\vdash^*$ é de fato uma relação pode-se facilmente que a igualdade entre derivações nada mais é do que a igualdade entre tuplas ordenadas.

Definição 19 (Linguagem de uma gramática) Dado uma gramática $G = \langle V, \Sigma, S, P\rangle$ a linguagem gerada por G, denotada por $\mathcal{L}(G)$, corresponde ao conjunto formado por todas as palavras sobre Σ que são deriváveis a partir do variável inicial da gramática, ou seja, $\mathcal{L}(G) = \{w \in \Sigma^* \mid S \vdash^* w\}$.

Exemplo 20 Não é difícil verificar que a gramática do Exemplo 16 gera a linguagem $\{w \in \{a\}^* \mid |w| = 2k, k \in \mathbb{N}\}$.

Agora o leitor pode ter notado que como gramática formais possuem um conjunto finito de regras, as linguagens por elas geradas nada mais são do que conjuntos indutivamente gerados.

1.5 Questionário

Questao 1 Dado o alfabeto $\Sigma = \{a, b, c\}$ e as palavras $u = aabcab, v = bbccabac$ e $w = ccbabbaaca$ determine:

(a). A palavra uv^r.

(b). A palavra $(w^r)^2u$.

(c). A palavra $((u^r)^2v^0)^rv$.

(d). A palavra uu^2v^rw.

(e). A palavra $((wuv)^r)^2u$.

(f). Determine o valor numérico da expressão $|w^3| + 2|v^2u| - |u|$.

(g). Determine o valor numérico da expressão $2|w^r| - |uv|$.

(h). Determine o valor numérico da expressão $|w^raaw| - |w|$.

(i). Determine o valor numérico da expressão $|uv^r| - 4$.

(j). Determine o valor numérico da expressão $\frac{|(w^r)^2u|}{2} - \frac{|u|}{6}$.

Questao 2 Demonstre que, se u é um prefixo de v, então $|u| \leq |v|$.

Questao 3 Demonstre para quaisquer palavras u e v e para todo $n \in \mathbb{N}$ as asserções a seguir.

(a). $|u^n| = n|u|$.

(b). $|(uv)^r| = |vu|$.

(c). Se $|u| = n$, então $n \leq |uv|$.

Questao 4 Considere a linguagem $L = \{\lambda, abb, a, abba\}$ e determine:

(a). $L^r - \{\lambda, a\}$.

(b). L^3.

(c). $PRE(L)$.

(d). $SUF(L^2)$.

(e). w tal que $|w| = max\Big(\{|w'| \mid w' \in L^3\}\Big)$.

Questao 5 Prove que para qualquer linguagem L e quaisquer $m, n \in \mathbb{N}$ as seguintes asserções.

(a). $(L^m)^n = L^{mn}$.

(b). $L^mL^n = L^{m+n}$.

(c). $(L^r)^n = (L^n)^r$.

(d). $\overline{L}^r = \overline{L^r}$.

(e). $PRE(L) = (SUF(L^r))^r$.

Questao 6 Dado duas linguagens quaisquer L_1 e L_2 demostre ou apresente um contra-exemplo para os seguintes enunciados:

(a). Se $L_1 \cap L_2 \neq \emptyset$, então $PRE(L_1) \cap PRE(L_2) = \emptyset$.

(b). Se $L_1 \subseteq L_2$, então $SUF(L_1) \cap SUF(L_2) = \emptyset$.

(c). Se $L_1 \subseteq L_2$, então $L_1^r \subseteq L_2^r$.

(d). Se $L_1 \subseteq L_2$, então para todo L tem-se que $LL_1 \subseteq LL_2$.

Questao 7 Demonstre ou refute o predicado $(\forall L \subseteq \Sigma^*)[(\forall n \in \mathbb{N})[\overline{L}^n = \overline{L^n}]]$.

Questao 8 Esboce uma linguagem L não trivial[a], para que a igualdade:

$$PRE(L) = (SUF(L))^r$$

seja verdadeira.

[a] Trivial aqui diz respeito a uma linguagem que não seja o próprio alfabeto ou o conjunto $\{\lambda\}$.

Parte II

Linguagens Regulares

Capítulo 2

Autômatos Finitos e suas Linguagens

"Eu acredito que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar. "

Alan M. Turing.

Como explicado em diversas obras tais como [6, 13, 15, 21], os autômatos finitos podem ser separados em dois tipos bem definidos, a saber, Autômato Finito Determinístico (AFD) e Autômato Finito Não-determinístico (AFN).

2.1 Autômato Finito Determinístico

Agora este documento inicia o estudo dos AFD apresentando sua forma algébrica equacional.

Definição 20 (Autômato Finito Determinístico) Um AFD é uma estrutura $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ onde: Q é um conjunto finito de estados, Σ é um alfabeto, $\delta : Q \times \Sigma \to Q$ é uma função total (chamada função de transição), $q_0 \in Q$ é um estado destacado (chamado estado inicial) e $F \subseteq Q$ é o conjunto de estados finais[a].

[a] Em algumas referências também é usado o termo conjunto de estados de aceitação [12].

Exemplo 21 A estrutura $A = \langle \{q_0, q_1\}, \{a\}, \delta, q_0, \{q_1\}\rangle$ onde a função de transição é definida por: $\delta(q_0, a) = q_1$ e $\delta(q_1, a) = q_0$, é um AFD.

Exemplo 22 A estrutura $B = \langle \{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_0\}\rangle$ onde a função de transição é definida por:
não é um AFD, pois $\delta(q_2, b)$ não está definido, e portanto, δ não é uma função total furando assim a definição de AFD.

Não é um fato claro por que usamos a letra Q para representar o conjunto de estados, e a letra minúscula q para simbolizar a um estado genérico, mas isso agora é o padrão notacional firmemente estabelecido nas principais obras da área [6, 13, 15].

Embora o estudo formal dos autômatos finitos tenha começado muito antes, sua formulação moderna foi estabelecida pelo artigo[1] do ano de 1959 escrito por Michael Rabin e Dana Scott [24]. Em tal artigo Rabin e Scott chamaram o conjunto de estados

[1] Esse artigo levou Michael Rabin e Dana Scoot a ganharem o Prêmio Turing.

de S, usaram a letra minúscula s para um estado genérico e chamaram o estado inicial de s_0.

Por outro lado, no artigo de 1936, que é o trabalho seminal da teoria da computação, Turing usou $q_1, q_2, \cdots, q_R$ para se referir aos estados (ou "m-configurações") de uma máquina de Turing genérica, não existe uma evidência forte que explique o motivo da escolha do q. Então o leitor não deve se preocupa se hora esse texto usar q e depois s para representar os estados, pois isso é apenas um conceito de notação e estética textual.

Atenção

As siglas AFD e AFN serão usado tanto para designar o singular quanto o plural, ficando a distinção a critério das sentenças envolvendo tais singlas.

A função de transição (δ) pode ser interpretada semanticamente como sendo o programa que o autômato executa, assim uma aplicação qualquer de δ é uma instrução do programa do autômato, por exemplo, a aplicação $\delta(q, x) = p$ significa que, o AFD muda do estado atual q para o estado p quando o mecanismo de leitura lê o símbolo x na memória.

Uma representação comum para os AFD é baseada no uso de grafos de transição [11]. Em um grafo de transição os vértices irão ser representados por círculos, que neste caso são usados para representar os estados do autômato, isto é, os círculos representam os elementos de Q. Cada aresta (q_i, q_j) são rotuladas por x representando assim a transição da forma $\delta(q_i, x) = q_j$. Por fim, os estados finais, isto é, cada $q \in F$ será representado por vértices desenhados como um círculo duplo em vez de um círculo simples e o estado inicial é marcado com uma seta.

Exemplo 23

A representação por grafo de transição do AFD descrito no Exemplo 21 corresponde a figura a seguir.

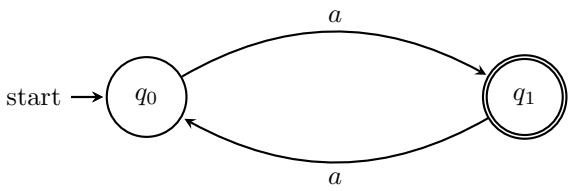


Figura 2.1: Representação visual do AFD no Exemplo 21.

Exemplo 24

O AFD $S = \langle \{s_0, s_1, s_2\}, \{0, 1\}, \delta, s_0, \emptyset \rangle$ onde a função de transição é definida como sendo: $\delta(s_0, 0) = s_1, \delta(s_1, 0) = s_2, \delta(s_2, 0) = s_1, \delta(s_0, 1) = s_2, \delta(s_1, 1) = s_1$ e $\delta(s_2, 1) = s_1$, é um AFD e pode ser representado pela Figura 2.2 a seguir.

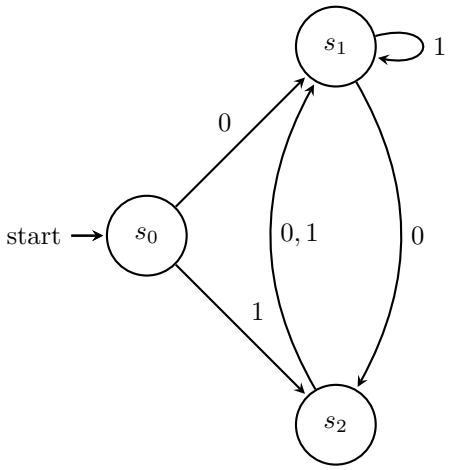


Figura 2.2: Representação visual do AFD S do Exemplo 24.

Pode-se agora então estender a função de transição, para que o autômato possa vir a processar palavras, em vez de apenas símbolos individuais.

Definição 21 (Função de Transição Estendida) Seja $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ um AFD a função δ é estendida para uma função $\widehat{\delta} : Q \times \Sigma^* \to Q$ usando recursividade como se segue.

$$\widehat{\delta}(q, \lambda) = q \tag{2.1}$$
$$\widehat{\delta}(q, wa) = \delta(\widehat{\delta}(q, w), a) \tag{2.2}$$

onde $q \in Q, a \in \Sigma$ e $w \in \Sigma^*$.

A partir da definição de função de transição estendida é definida a noção de computação para os AFD, tal conceito é formalizado a seguir.

Definição 22 (Computação em AFD) Seja $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ um AFD e seja $w \in \Sigma^*$ uma computação de w em A corresponde a aplicação $\widehat{\delta}(q_0, w)$.

Note que a definição de computação em AFD pode ser interpretada como sendo a resposta ao seguinte questionamento: "Em que estado o autômato (ou a máquina) estará após iniciar o processamento no estado inicial e ter lido todos os símbolos da palavra de entrada w?"

Exemplo 25 Considere o AFD do Exemplo 21 e a palavra de entrada *aaaa* tem-se que a computação desta palavra corresponde a:

$$\begin{aligned}
\widehat{\delta}(q_0, aaaa) &= \delta(\widehat{\delta}(q_0, aaa), a)\\
&= \delta(\delta(\widehat{\delta}(q_0, aa), a), a)\\
&= \delta(\delta(\delta(\widehat{\delta}(q_0, a), a), a), a)\\
&= \delta(\delta(\delta(\delta(\widehat{\delta}(q_0, \lambda), a), a), a), a)\\
&= \delta(\delta(\delta(\delta(q_0, a), a), a), a)\\
&= \delta(\delta(\delta(q_1, a), a), a)\\
&= \delta(\delta(q_0, a), a)\\
&= \delta(q_1, a)\\
&= q_0
\end{aligned}$$

Exemplo 26 Considere o AFD do Exemplo 24 e a palavra de entrada 0101 tem-se que a computação desta palavra corresponde a:

$$\begin{aligned}
\widehat{\delta}(s_0, 0101) &= \delta(\widehat{\delta}(s_0, 010), 1)\\
&= \delta(\delta(\widehat{\delta}(s_0, 01), 0), 1)\\
&= \delta(\delta(\delta(\widehat{\delta}(s_0, 0), 1), 0), 1)\\
&= \delta(\delta(\delta(\delta(\widehat{\delta}(s_0, \lambda), 0), 1), 0), 1)\\
&= \delta(\delta(\delta(\delta(s_0, 0), 1), 0), 1)\\
&= \delta(\delta(\delta(s_1, 1), 0), 1)\\
&= \delta(\delta(s_1, 0), 1)\\
&= \delta(s_2, 1)\\
&= s_1
\end{aligned}$$

De pose da definição de computação pode-se formalizar o conceito de reconhecimento (ou aceitação) de palavras nos AFD.

Definição 23 (Reconhecimento de palavras em AFD) [6] Sejam $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ um AFD e seja $w \in \Sigma^*$. A palavra w é dita aceita (reconhece ou computada) por A sempre que $\widehat{\delta}(q_0, w) \in F$ e é rejeitada por A em qualquer outro caso.

É fácil perceber que $\widehat{\delta}(q_0, w) \in F$ com $w = a_1 a_2 \cdots a_n$ se, e somente se, existir uma sequência finita de estados $(q_i)_{i\in I}$ tal que,

$$\delta(q_0, a_1) = q_{i_1}, \delta(q_{i_1}, a_2) = q_{i_2}, \cdots, \delta(q_{i_{n-1}}, a_n) = q_{i_n}$$

com $q_n \in F$, sendo uma sequencia de números naturais e $i_1, i_2, i_{n-1}, i_n \in I$. O leitor pode notar que em particular tem-se que $\widehat{\delta}(q_0, \lambda) \in F$ se, e somente se, $q_0 \in F$.

Exemplo 27 Considerando os Exemplos 25 e 26 tem-se que a palavra $aaaa$ não é aceita pelo AFD do Exemplo 25, uma vez que, $q_0 \notin F$. Já a palavra 0101 também não é aceita pelo AFD do Exemplo 26, uma vez que, $s_1 \notin F$, de fato o leitor atento pode notar que o AFD do Exemplo 26 não aceita qualquer palavra de entrada pois $F = \emptyset$.

Exemplo 28 Considere o AFD representado pelo grafo de transições abaixo:

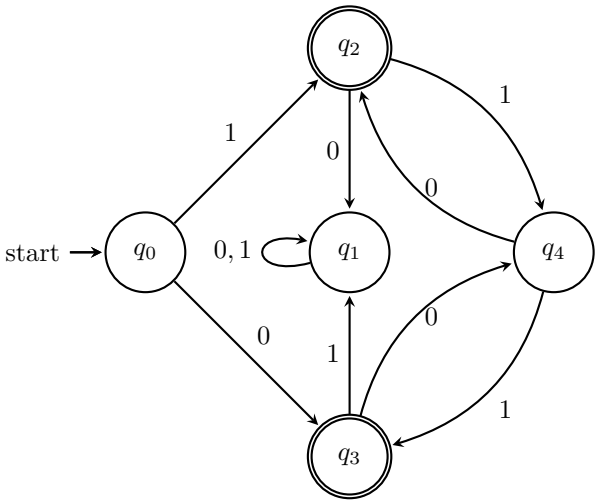


Figura 2.3: Um AFD com dois estados finais.

Por indução sobre o tamanho das palavras é fácil mostrar que este AFD reconhece palavras das forma $1(10)^n$ e $1(10)^n$ com $n \in \mathbb{N}$.

Tendo definido precisamente as noções de AFD e de computação em AFD, agora é possível definir formalmente a ideia de linguagem reconhecida (ou computada) por um AFD.

Definição 24 (Linguagem de um AFD) Seja $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ um AFD a linguagem reconhecida (ou computada) por A, denotada por $\mathcal{L}(A)$, corresponde ao conjunto de todas as palavras aceitas por A, formalmente tem-se que:

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \widehat{\delta}(q_0, w) \in F\} \tag{2.3}$$

Utilizando a definição acima o leitor deve ser capaz de perceber que se um AFD reconhece uma linguagem $L \subseteq \Sigma^*$, então ele para em estados finais apenas para as palavras $w \in L$. Em outra palavra para mostrar que uma linguagem L é a linguagem de um AFD A, deve-se provar que $L = \mathcal{L}(A)$, ou seja, deve-se provar que $w \in L \iff w \in \mathcal{L}(A)$, em geral quando L é infinito tal prova é por indução.

Exemplo 29 A seguir você encontrará a prova de que a linguagem $L = \{bba^{2n} \mid n \in \mathbb{N}\}$ é reconhecida pelo AFD A_1 na Figura 2.4 a seguir.

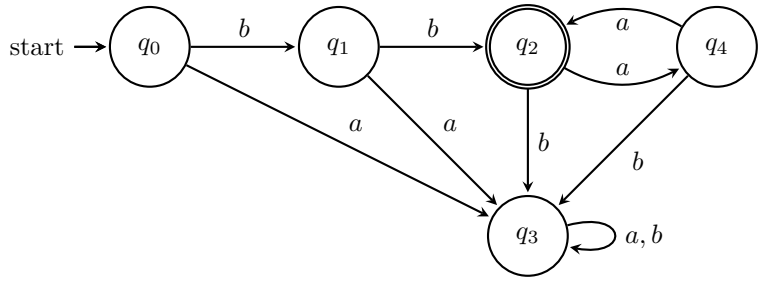


Figura 2.4: AFD A_1 que reconhece a linguagem $\{bba^{2n} \mid n \in \mathbb{N}\}$.

Prova

($\Rightarrow$) Suponha que $w \in L$ assim $w = bba^{2n}$ e por indução sobre o tamanho das palavras tem-se que,

(B)ase: Quando $n = 0$ vale que $w = bba^{2\cdot 0}$ e usando a definição do AFD tem-se que,

$$\widehat{\delta}(q_0, bba^{2\cdot 0}) = \widehat{\delta}(q_0, bb) = \delta(\widehat{\delta}(q_0, b), b) = \delta(\delta(\widehat{\delta}(q_0, \lambda), b), b) = q_2$$

como $q_2 \in F$ tem-se que $bb \in \mathcal{L}(A_1)$.

(H)ipótese indutiva: Suponha que para todo $n \in \mathbb{N}$ tem-se que $\widehat{\delta}(q_0, bba^{2n}) \in F$, ou seja, $\widehat{\delta}(q_0, bba^{2n}) = q_2$.

(P)asso indutivo: Dado $w = bba^{2(n+1)}$ tem-se que

$$\begin{aligned}
\widehat{\delta}(q_0, bba^{2(n+1)}) &= \widehat{\delta}(q_0, bba^{2n+2}) \\
&= \widehat{\delta}(q_0, bba^{2n}aa) \\
&= \delta(\delta(\widehat{\delta}(q_0, bba^{2n}), a), a) \\
&\overset{\textbf{(HI)}}{=} \delta(\delta(q_2, a), a) \\
&= \delta(q_3, a) \\
&= q_2
\end{aligned}$$

Logo, por **(B)**, **(H)** e **(P)** tem-se que $\widehat{\delta}(q_0, bba^{2n}) \in \mathcal{L}(A_1)$ para qualquer que seja $n \in \mathbb{N}$.

($\Leftarrow$) Suponha que $w \in \mathcal{L}(A_1)$, assim pela definição do AFD A_1 tem-se que $\widehat{\delta}(q_0, w) = q_2$, entretanto, pela definição de δ (ver Figura 2.4) tem-se que q_2 só é acessado pelas transições $\delta(q_1, b)$ e $\delta(q_4, a)$, ou seja, $w = w_1a$ ou $w = w_2b$ com $w_1, w_2 \in \Sigma^*$. Agora analisando cada possibilidade em separado tem-se que:

- Para realizar o acesso via q_1 é necessário obviamente chegar em q_1 e isso só é possível a partir da transição $\delta(q_0, b)$, logo o acesso a q_2 via q_1 só é permitido para palavras com o prefixo bb, agora como toda palavra é prefixo de si mesmo isso já garante que $bb \in \mathcal{L}(A_1)$.
- Já o acesso via q_4 só é permitido pela transição $\delta(q_2, a)$ e como visto no caso anterior tem-se que o estado q_2 só pode ser acessado por palavras com prefixo bb, note porém, que as transições $\delta(q_2, a) = q_4$ e $\delta(q_4, a) = q_2$ formam um *loop* e assim pode-se concluir que o acesso a q_2 via q_4 obrigatoriamente é realizado por palavras da forma bba^{2n} com $n \geq 1$.

Note que a palavra bb pode ser escrita como sendo bba^0, portanto, pelas duas análises anteriores pode-se concluir que se $\widehat{\delta}(q_0, w) = q_2$, então $w = bba^{2n}$ com $n \in \mathbb{N}$, e portanto, $w \in L$, completando assim a prova. □

Exemplo 30 O AFD A do Exemplo 21 reconhece a linguagem $L = \{a^{2n+1} \mid n \in \mathbb{N}\}$.

Prova

($\Rightarrow$) Suponha que $w \in L$ assim $w = a^{2n+1}$, agora por indução sobre o tamanho das palavras tem-se que,

(B)ase: Quando $n = 0$ vale a igualdade $w = a^{2\cdot 0+1}$, agora usando a definição do AFD A tem-se que,

$$\widehat{\delta}(q_0, a^{2\cdot 0+1}) = \widehat{\delta}(q_0, a^1) = \delta(\widehat{\delta}(q_0, \lambda), a) = \delta(q_0, a) = q_1$$

e como $q_1 \in F$ tem-se que $a^{2\cdot 0+1} \in \mathcal{L}(A)$, ou seja, $w \in \mathcal{L}(A)$.

(H)ipótese indutiva: Suponha que para todo $n \in \mathbb{N}$ tem-se que $\widehat{\delta}(q_0, a^{2n+1}) \in F$, ou seja, $\widehat{\delta}(q_0, a^{2n+1}) = q_1$.

(P)asso indutivo: Dado $w = a^{2(n+1)+1}$ tem-se que,

$$\begin{aligned}
\widehat{\delta}(q_0, a^{2(n+1)+1}) &= \widehat{\delta}(q_0, a^{2n+1+2}) \\
&= \widehat{\delta}(q_0, a^{2n+1}aa) \\
&= \delta(\delta(\widehat{\delta}(q_0, a^{2n+1}), a), a) \\
&\overset{\textbf{(HI)}}{=} \delta(\delta(q_1, a), a) \\
&= \delta(q_0, a) \\
&= q_1
\end{aligned}$$

Logo, por **(B)**, **(H)** e **(P)** tem-se que $\widehat{\delta}(q_0, a^{2n+1}) \in \mathcal{L}(A_1)$ para qualquer que seja $n \in \mathbb{N}$.

($\Leftarrow$) A volta fica como exercício argumentativo ao leitor. □

Pode-se agora formalizar a primeira das classes de linguagens sendo esta a classe das linguagens regulares, tal classe foi primeiramente definida por Kleene em seu trabalho [14], entretanto, em tal ocasião tais linguagens foram chamadas de eventos regulares, como será visto é momentos futuros nesse manuscrito a classe das linguagens regulares é aquela que possui o menor nível complexidade computacional.

Definição 25 (Linguagens Regulares) Uma linguagem L qualquer é dita ser regular se, e somente se, existe um AFD A tal que $L = \mathcal{L}(A)$. A classe de todas as linguagens regulares é denotada por $\mathcal{L}_{Reg}$.

Referências Bibliográficas

[1] J. M. Abe and N. Papavero. *Teoria Intuitiva dos Conjuntos*. MAKRON *Books*, 1991.

[2] A. V. AHO, M. S. LAM, R. SETHI, and J. D. ULLMAN. *Compiladores: Princípios, Técnicas e ferramentas*. Editora Pearson, 2 edition, 2007.

[3] J. Avigad. Handbook of proof theory. In *Studies in Logic and the Foundations of Mathematics, ch*. Citeseer, 1998.

[4] M. Ayala-Rincón and F. L. C. de Moura. *Fundamentos da Programação Lógica e Funcional – O princípio de Resolução e a Teoria de Reescrita*. Editora UnB, 2014.

[5] B. Bedregal and B. M. Acióly. Introdução à lógica clássica para a ciência da computação. Notas de Aula, 2007.

[6] B. Bedregal, B. M. Acióly, and A. Lyra. *Introdução à Teoria da Computação: Linguagens Formais, Autômatos e Computabilidade*. Editora UnP, Natal, 2010.

[7] S. R. Buss. *Handbook of proof theory*. Elsevier, 1998.

[8] N. Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

[9] K. Cooper and L. Torczon. *Construindo Compiladores*, volume 1. Elsevier Brasil, 2017.

[10] V. S. Costa. Linguagens Lineares Fuzzy. Master's thesis, Programa de Pós-graduação em Sistemas e Computação, Universidade Federal do Rio Grande do Norte, UFRN, Natal, RN, 2016.

[11] V. S. Costa. *Autômatos Fuzzy Hesitantes Típicos: Teoria e Aplicações*. PhD thesis, Programa de Pós-graduação em Sistemas e Computação, Universidade Federal do Rio Grande do Norte, UFRN, Natal, RN, 2020.

[12] C. De la Higuera. *Grammatical inference: Learning Automata and Grammars*. Cambridge University Press, London, 2010.

[13] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Pearson Education India, USA, 3ł edition, 2008.

[14] S. C. Kleene. Representation of events in nerve nets and finite automata. Technical report, Rand Project Air Force Santa Monica CA, 1951.

[15] P. Linz. *An Introduction to Formal Languages and Automata*. Jones & Bartlett Learning, New York, 2006.

[16] S. Lipschutz. *Teoria dos Conjuntos*. McGraw-Hill do Brasil - LTDA/MEC, 1978.

[17] S. Lipschutz and M. Lipson. *Matemática Discreta*. Bookman Editora, 2013. Coleção Schaum.

[18] J. P. Martins. *Lógica e Raciocínio*. College Publications, 2014.

[19] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[20] G. H. Mealy. A method for synthesizing sequential circuits. *The Bell System Technical Journal*, 34(5):1045–1079, 1955.

[21] P. B. Menezes. *Linguagens Formais e Autômatos*. Sagra-Dcluzzato, 1998.

[22] E. F. Moore. Gedanken-experiments on sequential machines. *Automata Studies*, 34:129–153, 1956.

[23] M. O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

[24] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[25] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265, 1937.