



Computação Formal

Um Compêndio dos Fundamentos Matemáticos da Computação

Autor: Valdigleis S. Costa

Instituição: Grupo ALiCIA

Data: 27 de março de 2022

Versão: 1.0

Bio: Informações



Este manuscrito não possui qualquer vínculo editorial.

Conteúdo

I	Ferramentas e Linguagem Básica	1
1	Conjuntos	2
1.1	Conjuntos e Elementos	2
1.2	Operações sobre conjuntos	6
1.3	Operações generalizadas	14
1.4	Partes e partições	15
	Questionário	16
2	Métodos de Demonstração	22
2.1	Introdução	22
2.2	Demonstrando Implicações	23
II	Lógica	26
3	Introdução à Lógica	27
3.1	O que é Lógica?	27
3.2	Um Pouco de História	28
3.3	Argumentos, Proposições e Predicados	30
3.4	Conectivos, Quantificadores e Negação	32
3.5	Representação simbólica	33
3.6	Lógica e Ciência da Computação	33
	Questionário	34
4	Lógica Proposicional	37
III	Álgebra Universal	38
IV	Teoria dos Números e Combinatória	39
V	Linguagens Formais, Autômatos e Computabilidade	40
5	Introdução	41
5.1	Sobre as Linguagens Formais	41
5.2	Sobre Autômatos Finitos	41
5.3	Noções Fundamentais	42
5.4	Sobre Gramática Formais	47
	Questionário	49

6	Linguagens Regulares	50
6.1	Autômato Finito Determinístico	50
6.2	Autômato Finito Não-determinístico	55
6.3	Autômatos Finitos Não-determinísticos com Movimentos Vazios	62
6.4	Teorema Myhill-Nerode e a Minimização de AFD	68
6.5	Máquinas de Mealy e Moore	76
6.6	A Notação Matricial	80
6.7	Expressões Regulares	80
6.8	Gramática Regulares	88
	Questionário	93
7	Linguagens Livres do Contexto	100
7.1	Gramática Livres do Contexto	100
7.2	Simplificação de Gramática Livres do Contexto	104
7.3	Sobre Algoritmos de Pertinência em GLC	119
7.4	Sobre GLC e Linguagens de Programação	119
7.5	Autômatos de Pilha	119
	Questionário	123
VI	λ-Cálculo	126
VII	Complexidade	127

I

FERRAMENTAS E LINGUAGEM BÁSICA

“-Comece pelo começo”, disse o Rei de maneira severa,
“-E continue até chegar ao fim, então pare!”.

LEWIS CARROLL, ALICE NO PAÍS DAS MARAVILHAS.

Capítulo Conjuntos

Tópicos

❑ Conjuntos e Elementos

❑ Operações Sobre Conjuntos

❑ Operações Generalizadas

❑ Partes e Partições

❑ Questionário

1.1 Conjuntos e Elementos

A ideia de conjunto é provavelmente o conceito mais fundamental compartilhado pelos mais diversos ramos da matemática. O primeiro grande estudioso que apresentou uma forma relativamente precisa para o conceito de conjunto foi o matemático alemão George Cantor (1845-1918) em seu seminal trabalho [15]. A seguir será apresentada uma tradução não literal da definição original de Cantor.

Definição 1.1 (Cantor)

Um **conjunto** A é uma **coleção** numa totalidade M de certos **objetos** distintos e bem-definidos n que fazem parte da nossa percepção ou pensamento, tais objetos são chamados de **elementos** de A .

Note que a definição apresentada por Cantor exige dois aspectos sobre a natureza dos elementos em um conjunto: (1) que eles sejam distintos entre si, ou seja, em um conjunto não poderá haver repetição de elementos e (2) os elementos devem ser bem-definidos.

A definição de Cantor permite que sejam criados conjuntos com qualquer coisa que o indivíduo racional possa pensar ou perceber pelos seus sentidos. Agora, entretanto, deve-se questionar o que significa dizer que algo é bem-definido? Uma resposta satisfatória para essa pergunta é dizer que algo é bem-definido se esse algo pode ser descrito sem ambiguidades.

É claro que qualquer coisa pode ser descrita a partir de suas propriedades, características ou atributos, sendo que essas propriedades sempre podem ser verificadas pelos sentidos no caso de objetos físicos, e sempre pode-se pensar e argumentar sobre elas no caso de objetos abstratos. Assim pode-se modificar um pouco a definição de Cantor para a definição que se segue.

Definição 1.2 (Definição de Cantor Modificada)

Um **conjunto** A é uma **coleção** numa totalidade M de certos **objetos** n distintos e que satisfazem certas propriedades, tais objetos são chamados de **elementos** de A .

Observação: A partir desse ponto será usado a nomenclatura discurso em vez de totalidade na especificação de conjuntos.

Note que a Definição 1.2 permite concluir que um conjunto pode ser visto como o agrupamento de entidades (os elementos) que satisfazem certas propriedades, ou ainda que, as propriedades definem os conjuntos. Para prosseguir nesse texto serão adotadas as convenções para a **linguagem da teoria dos ingênuos dos conjuntos** apresentadas pelas definições que se seguem.

Definição 1.3 (Notações Básicas)

As letras maiúsculas do alfabeto latino $A, B, \dots, M, N, \dots, Z$ como e sem indexação serão usadas como variáveis^a para representar conjuntos e as letras minúsculas $a, b, \dots, m, n, \dots, z$ como e sem indexação serão usadas como meta-variáveis para representar elementos.

^aFormalmente o termo meta-variável é usado para designar símbolos (ou palavras) de uma linguagem responsáveis por representar um tipo dentro da Teoria dos Tipos [54], aqui por outro lado, o termo é usado apenas para indicar que os símbolos representam outras entidades (conjuntos ou elementos), ou seja, tais símbolos são “apelidos” para certas entidades.



Como dito anteriormente uma propriedade P é responsável por definir um conjunto, pois todos os elementos no conjunto devem satisfazer (ou possuir) tal propriedade. Tendo isso em mente pode-se introduzir a definição a seguir.

Definição 1.4 (Notação compactada)

Um conjunto A definido por alguma propriedade P é representada na **forma compacta** como:

$$A = \{x \mid P\} \quad (1.1)$$



Observação: Na notação compacta $A = \{x \mid P\}$ o símbolo A é chamado de rótulo do conjunto, e a parte $\{x \mid P\}$ será chamada neste manuscrito de forma estrutural do conjunto. Sendo que durante este manuscrito pode ser referenciando conjuntos apenas por seus rótulos e outras vezes usando apenas suas formas estruturais.

Observação: A notação compacta $A = \{x \mid P\}$ pode ser lida como: “ A é o conjunto de todos os x tal que P é satisfeita por x ”.

Exemplo 1.1 Os seguintes conjuntos estão bem representados na notação compacta.

- (a) $X = \{a \mid a \text{ é uma cidade do Brasil}\}.$
- (b) $K = \{m \mid m \text{ é um animal mamífero}\}.$
- (c) $L = \{x \mid 0 \leq x < 10 \text{ e } x \text{ é um número ímpar}\}.$
- (d) $C = \{b \mid b \text{ é uma vogal}\}.$

Para continuar o desenvolvendo da linguagem da teoria dos conjuntos, é conveniente relembrar ao leitor os símbolos usados como rótulos para representar os conjuntos numéricos mais importantes da matemática.

Definição 1.5 (Símbolos dos conjuntos numéricos)

O conjunto dos números naturais^a, inteiros, racionais, irracionais, reais e complexos são representados respectivamente por \mathbb{N} , \mathbb{Z} , \mathbb{Q} , \mathbb{I} , \mathbb{R} e \mathbb{C} .

^aNeste manuscrito é considerado que o conjuntos dos naturais corresponde ao conjunto $\{0, 1, 2, \dots\}$.



Observação: Neste manuscrito \mathbb{N}_* , \mathbb{Z}_* , \mathbb{Q}_* e \mathbb{R}_* irão denotar receptivamente o conjunto dos naturais, inteiros, racionais e reais não nulos. Já \mathbb{Z}^+ , \mathbb{Q}^+ e \mathbb{R}^+ irão denotar receptivamente o conjunto dos inteiros, racionais e reais positivos. E por fim, \mathbb{Z}^- , \mathbb{Q}^- e \mathbb{R}^- irão denotar receptivamente o conjunto dos inteiros, racionais e reais negativos.

Seguindo com o desenvolvimento da teoria dos conjuntos a definição a seguir estabelece um relacionamento (ou relação) de pertinência entre os conjuntos e os elementos do discurso.

Definição 1.6 (Pertinência)

Seja A um conjunto definido sobre um discurso M por uma propriedade P e seja x um elemento do discurso. Se o elemento x possui (ou satisfaz) a propriedade P , então é dito que x pertence a A , denotado por $x \in A$. Caso x não possui (ou não satisfaça) a propriedade P , então é dito que x não pertence a A , denotado por $x \notin A$.



Observação: Quando $x \in A$, em alguns textos como em [36] é comum o uso de expressões como: “ A possui x ” ou que “ x faz parte de A ”, durante este manuscrito possa ser que uma dessas (ou ambas) expressões sejam usadas, além da expressão padrão x pertence a A .

Exemplo 1.2 Seja A o conjunto definido sobre a propriedade “é professor de Ciência da Computação na univasf” tem-se que o professor Rodrigo $\in A$. Já para os professores Regivan e Benjamin tem-se que Regivan, Benjamin $\notin A$.

Exemplo 1.3 Seja A_1 o conjunto definido pela propriedade “Clubes da primeira divisão do campeonato brasileira de futebol do ano 2021” tem-se então que Vasco $\notin A_1$.

Há casos entretanto, que a notação compacta é descartada e assim os conjuntos podem ser escritos simplesmente listando seus elementos entre as chaves da forma estrutural, isso em geral acontece quando o conjunto é finito¹ e possui um número não muito grande de elementos.

Exemplo 1.4 A seguir são listados alguns conjuntos finitos escritos descartando a notação compacta.

- (a) O conjunto das vogais pode ser representado como $A = \{a, e, i, o, u\}$.
- (b) O conjunto das siglas dos estados nordestinos pode ser escrito como $E = \{RN, PE, PB, MA, CE, SE, AL, BA, PI\}$.
- (c) O conjunto dos naturais menores que 10 é escrito como $N_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Observação: Quando se optar por escrever um conjunto finito apenas listando os seus elementos entre chaves a ordem com que os elementos aparecem não importa, assim tem-se que os conjuntos $\{a, e, i, o, u\}$ e $\{e, u, i, a, o\}$ são na verdade o mesmo conjunto.

Note que a relação de pertinência apresentada anteriormente (Definição 1.6) relaciona elementos e conjuntos, existe também uma relação extremamente fundamental dentro da teoria dos conjuntos que é definida entre dois conjuntos. Essa relação entre conjuntos recebe o nome de **relação de inclusão**, entretanto, como dito em [36], é comum que quando um conjunto A estiver relacionado com um conjunto B pela relação de inclusão, se usar a expressão “ A é subconjunto de B ”, em vez de, “ A está incluído em B ”. A seguir é apresentado formalmente esta relação.

Definição 1.7 (Relação de inclusão)

Dado dois conjuntos A e B quaisquer, é dito que A é subconjunto de B , denotado por $A \subseteq B$, quando todo $x \in A$ é tal que $x \in B$.



Exemplo 1.5 Dado o conjunto \mathbb{Z} tem-se que o conjunto $N = \{x \mid x \in \mathbb{Z} \text{ e } x = 2k \text{ para algum } k \in \mathbb{Z}\}$ é claramente um subconjunto de \mathbb{Z} pois todo número par é também um número inteiro.

Exemplo 1.6 As seguintes relações de inclusão se verificam:

- (a) $\{a, e, u\} \subseteq \{a, e, o, i, u\}$.
- (b) $\{x \mid x \text{ é uma cidade do PE}\} \subseteq \{x \mid x \text{ é uma cidade do Brasil}\}$.

¹Por hora o leitor deve considerar que um conjunto finito é aquele que o leitor poderia contar o número de elementos, em capítulos futuros serão formalizados os conceitos de conjuntos finitos e infinitos.

- (c) $\{x \mid x = 2k \text{ para algum } k \in \mathbb{N}\} \subseteq \mathbb{N}$.
 (d) $\{\text{Brasil}\} \subseteq \{x \mid x \text{ é um país do continente americano}\}$

No caso de existir um $x \in A$ tal que $x \notin B$ é dito que A não é subconjunto de B , e isto é denotado como $A \not\subseteq B$.

Exemplo 1.7 Seja $A = \{-1, 0, 1\}$ tem-se que $A \not\subseteq \mathbb{N}$.

Existe também a possibilidade de todos os elementos de A serem elementos de B , mas que B possua outros elementos que não fazem parte de A , nesse caso é dito que A é um **subconjunto próprio** de B , e isto é denotado como $A \subset B$.

Exemplo 1.8 As seguintes relações de inclusão se verificam:

- (a) $\{1, 2\} \subset \mathbb{R}$.
 (b) $\{x \mid x \text{ é uma cidade do PE}\} \subset \{x \mid x \text{ é uma cidade do Brasil}\}$.
 (c) $\mathbb{Z}_+ \subset \mathbb{Z}$.

Observação: Note que todo subconjunto A de um conjunto B pode ser visto como um conjunto construído sobre os elementos de B que satisfazem uma certa propriedade **P**, isto é, tem-se que todo subconjunto A é um conjunto da seguinte forma:

$$A = \{x \mid x \in B \text{ e } x \text{ satisfaz } \mathbf{P}\}$$

também é possível encontrar a notação:

$$A = \{x \in B \mid x \text{ satisfaz } \mathbf{P}\}$$

sempre que possível esse manuscrito irá adotar a segunda notação.

Usando a ideia de subconjunto pode-se como apresentado na literatura em obras como [2, 28, 36] introduzir a ideia de igualdade entre conjuntos, esta noção é apresentada formalmente como se segue.

Definição 1.8 (Igualdade de conjuntos)

[2] Dois conjuntos A e B são iguais, denotado por $A = B$, se e somente se, $A \subseteq B$ e $B \subseteq A$.



Teorema 1.1 (Teorema da igualdade)

[2] Sejam A, B e C conjuntos quaisquer. Então:

1. $A = A$.
2. Se $A = B$, então $B = A$.
3. Se $A = B$ e $B = C$, então $A = C$.



Dentro da teoria dos conjuntos alguns conjuntos possuem tanta importância e destaque que eles recebem nomes e símbolos próprios.

Definição 1.9 (Conjunto Universo)

O conjunto universo, ou universo do discurso, denotado por \mathbb{U} , é um conjunto que possui todos os elementos sobre os quais se “fala”.

^aO termo fala aqui diz respeito ao ato pensar ou argumentar sobre os objetos.



O universo do discurso não é único, de fato o mesmo muda em função sobre o que se está “discursando”, por exemplo, pode-se pensar em um universo do discurso para falar sobre números, carros, pessoas, animais, palavras, times de futebol e etc.

Definição 1.10 (Conjunto vazio)

O conjunto vazio, denotado por \emptyset , corresponde a um conjunto que não possui nenhum elemento.



Uma propriedade interessante sobre o conjunto vazio é apresentada a seguir, tal propriedade garante que o conjunto vazio está presente em qualquer outro conjunto existente.

Teorema 1.2

Para todo conjunto A tem-se que $\emptyset \subseteq A$.



Demonstração Suponha por absurdo que existe um conjunto A tal que $\emptyset \not\subseteq A$, assim por definição existe pelo menos um $x \in \emptyset$ tal que $x \notin A$, mas isto é um absurdo já que o vazio não possui elementos, e portanto, a afirmação que $\emptyset \not\subseteq A$ é falsa, logo, $\emptyset \subseteq A$ é verdadeiro para qualquer que seja o A . ■



Nota: Neste manuscrito ao final das demonstrações será sempre colocado o símbolo ■, tal símbolo é conhecido como túmulo de Halmos², este simbolo será usado para substituir a notação q.e.d. (“quod erat demonstrandum”) usando por outras fontes bibliográficas para marcar o ponto de finalização de uma demonstração.

Agora que foram apresentados os conjuntos universo e vazio, é conveniente comentar sobre uma situação específica da teoria dos conjuntos como apresentada até aqui. Pelo que foi apresentado até agora já se sabe que os itens em um conjunto são chamados de elementos, entretanto, não existe qualquer restrição, além de ser bem definido, para a natureza (ou tipo) dos elementos em um conjunto. Isso possibilita que seja possível definir por exemplo um conjunto de conjuntos, isto é, um conjunto em que os elementos são também conjuntos.

Definição 1.11 (Família)

Um conjunto A cujo os elementos são todos conjuntos, isto é, um conjunto da forma $A = \{x \mid x \text{ é um conjunto}\}$, é chamado de **família de conjuntos**.



Exemplo 1.9 Os conjuntos:

$$A_1 = \{\mathbb{Z}_+^*, \mathbb{Z}_-, \{\pi, \sqrt{-1}\}\} \text{ e } A_2 = \{\{a, b\}, \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}, \mathbb{R}\}$$

são ambas famílias.

Observação: Além do termo família algumas obras como [36] também usam a nomenclatura classe, neste manuscrito só será usado o termo classe em situações bem específicas como por exemplo, as classes de equivalência em um espaço quociente.

1.2 Operações sobre conjuntos

Seguindo a mesma organização de conteúdo apresentada em [37], pode-se agora introduzir uma série de operações conjuntistas, isto é, operações que agem diretamente sobre conjuntos de “entrada” produzindo como “saída” novos conjuntos.

Definição 1.12 (União de conjuntos)

Sejam A e B dois conjuntos quaisquer, a união de A com B , denotada por $A \cup B$, corresponde ao seguinte conjunto.

$$A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$$



²Em inglês esse símbolo é conhecido como *tombstone*, e tal símbolo foi usado para marcar o final de uma demonstração inicialmente pelo matemático Paul Halmos (1916-2006).

Exemplo 1.10 Dados os dois conjuntos $A = \{x \in \mathbb{N} \mid x = 2i \text{ para algum } i \in \mathbb{N}\}$ e $B = \{x \in \mathbb{N} \mid x = 2j + 1 \text{ para algum } j \in \mathbb{N}\}$ tem-se que $A \cup B = \mathbb{N}$.

Exemplo 1.11 Seja $N = \{1, 2, 3, 6\}$ e $L = \{4, 6\}$ tem-se que $N \cup L = \{1, 2, 3, 4, 6\}$.

Como apontado em [36] alguns livros usam a notação $A + B$ para representar a união, é comum nesse caso não usar a nomenclatura união, em vez disso, é usado o termo soma de conjunto, entretanto, trata-se da mesma operação de união apresentada na definição anterior.

Definição 1.13 (Interseção de conjuntos)

Sejam A e B dois conjuntos quaisquer, a interseção de A com B , denotada por $A \cap B$, corresponde ao seguinte conjunto.

$$A \cap B = \{x \mid x \in A \text{ e } x \in B\}$$

Exemplo 1.12 Dado $A_1 = \{x \in \mathbb{N} \mid x \text{ é múltiplo de } 2\}$ e $A_2 = \{x \in \mathbb{N} \mid x \text{ é múltiplo de } 3\}$ tem-se que $A_1 \cap A_2 = \{x \in \mathbb{N} \mid x \text{ é múltiplo de } 6\}$.

Exemplo 1.13 Seja $A = \{1, 2, 3\}$, $B = \{2, 3, 4, 5\}$ e $C = \{5\}$ tem-se que:

- (a) $A \cap B = \{2, 3\}$.
- (b) $A \cap C = \emptyset$.
- (c) $B \cap C = \{5\}$.
- (d) $B \cap B = \{2, 3, 4, 5\} = B$.

Com respeito as propriedades equacionais das operações de união e interseção tem-se como exposto em [37] os seguintes resultados para todo A, B e C .

identificador	None	União	Interseção
p_1	Idempotência	$A \cup A = A$	$A \cap A = A$
p_2	Comutatividade	$A \cup B = B \cup A$	$A \cap B = B \cap A$
p_3	Associatividade	$A \cup (B \cup C) = (A \cup B) \cup C$	$A \cap (B \cap C) = (A \cap B) \cap C$
p_4	Distributividade	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
p_5	Neutralidade	$A \cup \emptyset = A$	$A \cap \mathbb{U} = A$
p_6	Absorção	$A \cup \mathbb{U} = \mathbb{U}$	$A \cap \emptyset = \emptyset$

Tabela 1.1: Tabela das propriedades das operações de união e interseção.

Além das propriedades apresentadas pela Tabela 1.1, a união e a interseção possuem propriedades ligadas a relação de inclusão.

Teorema 1.3

Para quaisquer conjuntos A e B tem-se que:

- i. $A \subseteq (A \cup B)$.
- ii. $(A \cap B) \subseteq A$

Demonstração Direta das Definições 1.7, 1.12 e 1.13. ■

A partir da definição de interseção é estabelecido um conceito de extrema valia para a teoria dos conjuntos e suas aplicações, tal conceito é o estado de disjunção entre dois conjuntos.

Definição 1.14 (Conjuntos disjuntos)

Dois conjuntos A e B são ditos disjuntos sempre que $A \cap B = \emptyset$.

Exemplo 1.14 Seja $A = \{1, 2, 3\}$, $B = \{2, 3, 5\}$ e $C = \{5\}$ tem-se que A e C são disjuntos, por outro lado, A e B não são disjuntos entre si, além disso, B e C também não são disjuntos entre si.

Definição 1.15 (Complemento de conjuntos)

Seja $A \subseteq \mathbb{U}$ para algum universo \mathbb{U} , o complemento de A , denotado por \overline{A} , corresponde ao seguinte conjunto:

$$\overline{A} = \{x \in \mathbb{U} \mid x \notin A\}$$



Exemplo 1.15 Dado $P = \{x \in \mathbb{Z} \mid x = 2k \text{ para algum } k \in \mathbb{Z}\}$ tem-se então o seguinte complemento $\overline{P} = \{x \in \mathbb{Z} \mid x = 2k + 1 \text{ para algum } k \in \mathbb{Z}\}$.

Exemplo 1.16 Dado um universo do discurso \mathbb{U} tem-se direto da definição que $\overline{\overline{\mathbb{U}}} = \emptyset$, e obviamente, $\overline{\emptyset} = \mathbb{U}$.

Teorema 1.4

Dado um conjunto A tem-se que:

- i. $A \cup \overline{A} = \mathbb{U}$.
- ii. $A \cap \overline{A} = \emptyset$.
- iii. $\overline{\overline{A}} = A$.



Demonstração Direta das Definições 1.12, 1.13 e 1.15. ■

Além das propriedades apresentadas no Teorema 1.4 o complemento também apresenta propriedades ligadas diretamente a união e a interseção, tais propriedades são uma versão conjuntistas das famosas leis De Morgan (ver [16, 37, 42]) muito conhecidas pelos estudiosos da área de lógica, a seguir são apresentadas as leis De Morgan para a linguagem teoria dos conjuntos.

$$\text{(DM1) Lei De Morgan 1ª forma: } \overline{(A \cup B)} = \overline{A} \cap \overline{B}$$

$$\text{(DM2) Lei De Morgan 2ª forma: } \overline{(A \cap B)} = \overline{A} \cup \overline{B}$$

Outra importante operação sobre conjuntos é a diferença definida formalmente como se segue.

Definição 1.16 (Diferença de conjuntos)

Dado dois conjuntos A e B , a diferença de A e B , denotado por $A - B$ corresponde ao seguinte conjunto:

$$A - B = \{x \in A \mid x \notin B\}$$



Exemplo 1.17 Dado os conjuntos $S = \{a, b, c, d\}$ e $T = \{f, b, g, d\}$ tem-se os seguintes conjuntos de diferença: $S - T = \{a, c\}$ e $T - S = \{f, g\}$.

Exemplo 1.18 Dado os conjuntos \mathbb{Z} e \mathbb{Z}_+^* tem-se que $\mathbb{Z} - \mathbb{Z}_+^* = \mathbb{Z}_-^*$.

Observação: Note que o Exemplo 1.17 mostra que a operação de diferença de conjuntos não é comutativa.

Teorema 1.5

Para todo A e B tem-se que:

- i. $A - B = A \cap \overline{B}$.
- ii. Se $B \subset A$, então $A - B = \overline{B}$.



Demonstração Dado os conjuntos A e B segue que:

- i. Por definição para todo $x \in A - B$ tem-se que $x \in A$ e $x \notin B$, mas isto só é possível se, e somente se, $x \in A$ e $x \in \overline{B}$, e por sua vez, isto só é possível se, e somente se, $x \in A \cap \overline{B}$, portanto, tem-se que $A - B = A \cap \overline{B}$.
- ii. Suponha que $B \subset A$, ou seja, todo $x \in B$ e tal que $x \in A$. Agora note que todo $x \in A - B$ é tal que $x \in A$ e $x \notin B$, e portanto, pela Definição 1.16 e pela hipótese de $B \subset A$ é claro que $A - B = \overline{B}$.



Os dois próximos teoremas estabelecem duas séries de diversas igualdades notáveis relacionadas a operação de diferença entre conjuntos.

Teorema 1.6

Sejam A e B subconjuntos de um universo \mathbb{U} , tem-se que:

- a. $A - \emptyset = A$ e $\emptyset - A = \emptyset$.
- b. $A - \mathbb{U} = \emptyset$ e $\mathbb{U} - A = \overline{A}$.
- c. $A - A = \emptyset$.
- d. $A - \overline{A} = A$.
- e. $\overline{(A - B)} = \overline{A} \cup B$.
- f. $A - B = \overline{B} - \overline{A}$.



Demonstração Para todas as equações a seguir suponha que A e B são subconjuntos de um universo \mathbb{U} assim segue que:

a.

$$\begin{aligned} A - \emptyset &\stackrel{\text{Teo. 1.5(i)}}{=} A \cap \overline{\emptyset} \\ &= A \cap \mathbb{U} \\ &\stackrel{\text{Tab. 1.1}(p_5)}{=} A \end{aligned}$$

e também tem-se que,

$$\begin{aligned} \emptyset - A &\stackrel{\text{Teo. 1.5(i)}}{=} \emptyset \cap \overline{A} \\ &\stackrel{\text{Tab. 1.1}(p_6)}{=} \emptyset \end{aligned}$$

b. A prova tem um raciocínio similar a demonstração do item anterior, assim será deixado como exercício ao leitor.

c. Trivial pela própria Definição 1.16.

d.

$$\begin{aligned} A - \overline{A} &\stackrel{\text{Teo. 1.5(i)}}{=} A \cap \overline{\overline{A}} \\ &\stackrel{\text{Teo. 1.4(iii)}}{=} A \cap A \\ &\stackrel{\text{Tab. 1.1}(p_1)}{=} A \end{aligned}$$

e.

$$\begin{aligned} \overline{(A - B)} &\stackrel{\text{Teo. 1.5(i)}}{=} \overline{(A \cap \overline{B})} \\ &\stackrel{\text{(DM2)}}{=} \overline{A} \cup \overline{\overline{B}} \\ &\stackrel{\text{Teo. 1.4(iii)}}{=} \overline{A} \cup B \end{aligned}$$

f.

$$\begin{aligned} A - B &\stackrel{\text{Teo. 1.5(i)}}{=} A \cap \overline{B} \\ &\stackrel{\text{Tab. 1.1}(p_2)}{=} \overline{B} \cap A \\ &\stackrel{\text{Teo. 1.4(iii)}}{=} \overline{B} \cap \overline{\overline{A}} \\ &\stackrel{\text{Teo. 1.5(i)}}{=} \overline{B} - \overline{A} \end{aligned}$$



Nota: Na demonstração do Teorema 1.6 apresentada anteriormente, algumas vezes foi escrito o símbolo de $=$ com um texto acima, isso é uma técnica comum na escrita de demonstrações matemáticas, o entendimento

que leitor precisa ter é que ao escrever $\stackrel{\kappa}{=}$ significa que a igualdade segue (ou é garantida) pela propriedade ou resultado κ . Durante este manuscrito em algumas demonstrações uma escrita similar irá aparecer para outros símbolos (implicações, bi-implicações e etc.) que serão introduzidos no decorrer deste manuscrito.

Teorema 1.7

Sejam A, B e C subconjuntos de um universo \mathbb{U} , tem-se que:

- a. $(A - B) - C = A - (B \cup C)$.
- b. $A - (B - C) = (A - B) \cup (A \cap C)$.
- c. $A \cup (B - C) = (A \cup B) - (C - A)$.
- d. $A \cap (B - C) = (A \cap B) - (A \cap C)$.
- e. $A - (B \cup C) = (A - B) \cap (A - C)$.
- f. $A - (B \cap C) = (A - B) \cup (A - C)$.
- g. $(A \cup B) - C = (A - C) \cup (B - C)$.
- h. $(A \cap B) - C = (A - C) \cap (B - C)$.
- i. $A - (A - B) = A \cap B$.
- j. $(A - B) - B = A - B$.



Demonstração Para todas as equações a seguir suponha que A, B e C são subconjuntos de um universo \mathbb{U} assim segue que:

a.

$$\begin{aligned}
 (A - B) - C &\stackrel{Teo. 1.5(i)}{=} (A \cap \overline{B}) \cap \overline{C} \\
 &\stackrel{Tab. 1.1(p_3)}{=} A \cap (\overline{B} \cap \overline{C}) \\
 &\stackrel{(DM1)}{=} A \cap \overline{(B \cup C)} \\
 &\stackrel{Teo. 1.5(i)}{=} A - (B \cup C)
 \end{aligned}$$

b.

$$\begin{aligned}
 A - (B - C) &\stackrel{Teo. 1.5(i)}{=} A \cap \overline{(B - C)} \\
 &\stackrel{Teo. 1.6(e)}{=} A \cap (\overline{B} \cup C) \\
 &\stackrel{Tab. 1.1(p_4)}{=} (A \cap \overline{B}) \cup (A \cap C) \\
 &\stackrel{Teo. 1.5(i)}{=} (A - B) \cup (A \cap C)
 \end{aligned}$$

c.

$$\begin{aligned}
 A \cup (B - C) &\stackrel{Teo. 1.5(i)}{=} A \cup (B \cap \overline{C}) \\
 &\stackrel{Tab. 1.1(p_4)}{=} (A \cup B) \cap (A \cup \overline{C}) \\
 &\stackrel{Tab. 1.1(p_2)}{=} (A \cup B) \cap (\overline{C} \cup A) \\
 &\stackrel{Teo. 1.4(iii)}{=} (A \cup B) \cap (\overline{C} \cup \overline{\overline{A}}) \\
 &\stackrel{(DM2)}{=} (A \cup B) \cap \overline{(C \cap \overline{A})} \\
 &\stackrel{Teo. 1.5(i)}{=} (A \cup B) - (C \cap \overline{A}) \\
 &\stackrel{Teo. 1.5(i)}{=} (A \cup B) - (C - A)
 \end{aligned}$$

d.

$$\begin{aligned}
A \cap (B - C) &\stackrel{Teo. 1.5(i)}{=} A \cap (B \cap \overline{C}) \\
&= \emptyset \cup (A \cap (B \cap \overline{C})) \\
&\stackrel{Tab. 1.1(p_2)}{=} \emptyset \cup ((A \cap B) \cap \overline{C}) \\
&\stackrel{Tab. 1.1(p_6)}{=} (\emptyset \cap B) \cup ((A \cap B) \cap \overline{C}) \\
&\stackrel{Teo. 1.4(ii)}{=} ((A \cap \overline{A}) \cap B) \cup ((A \cap B) \cap \overline{C}) \\
&\stackrel{Tab. 1.1(p_2, p_3)}{=} ((A \cap B) \cap \overline{A}) \cup ((A \cap B) \cap \overline{C}) \\
&\stackrel{Tab. 1.1(p_4)}{=} (A \cap B) \cap (\overline{A} \cup \overline{C}) \\
&\stackrel{(DM2)}{=} (A \cap B) \cap \overline{(A \cap C)} \\
&\stackrel{Teo. 1.5(i)}{=} (A \cap B) - (A \cap C)
\end{aligned}$$

e.

$$\begin{aligned}
A - (B \cup C) &\stackrel{Teo. 1.5(i)}{=} A \cap \overline{(B \cup C)} \\
&\stackrel{(DM1)}{=} A \cap (\overline{B} \cap \overline{C}) \\
&\stackrel{Tab. 1.1(p_1)}{=} (A \cap A) \cap (\overline{B} \cap \overline{C}) \\
&\stackrel{Tab. 1.1(p_3)}{=} ((A \cap A) \cap \overline{B}) \cap \overline{C} \\
&\stackrel{Tab. 1.1(p_2, p_3)}{=} ((A \cap \overline{B}) \cap A) \cap \overline{C} \\
&\stackrel{Tab. 1.1(p_3)}{=} (A \cap \overline{B}) \cap (A \cap \overline{C}) \\
&\stackrel{Teo. 1.5(i)}{=} (A - B) \cap (A - C)
\end{aligned}$$

f.

$$\begin{aligned}
A - (B \cap C) &\stackrel{Teo. 1.5(i)}{=} A \cap \overline{(B \cap C)} \\
&\stackrel{(DM2)}{=} A \cap (\overline{B} \cup \overline{C}) \\
&\stackrel{Tab. 1.1(p_4)}{=} (A \cap \overline{B}) \cup (A \cap \overline{C}) \\
&\stackrel{Teo. 1.5(i)}{=} (A - B) \cup (A - C)
\end{aligned}$$

g.

$$\begin{aligned}
(A \cup B) - C &\stackrel{Teo. 1.5(i)}{=} (A \cup B) \cap \overline{C} \\
&\stackrel{Tab. 1.1(p_4)}{=} (A \cap \overline{C}) \cup (B \cap \overline{C}) \\
&\stackrel{Teo. 1.5(i)}{=} (A - C) \cup (B - C)
\end{aligned}$$

h.

$$\begin{aligned}
(A \cap B) - C &\stackrel{Teo. 1.5(i)}{=} (A \cap B) \cap \overline{C} \\
&\stackrel{Tab. 1.1(p_4)}{=} (A \cap B) \cap (\overline{C} \cap \overline{C}) \\
&\stackrel{Tab. 1.1(p_2, p_3)}{=} (A \cap \overline{C}) \cap (B \cap \overline{C}) \\
&\stackrel{Teo. 1.5(i)}{=} (A - C) \cap (B - C)
\end{aligned}$$

i.

$$\begin{aligned}
A - (A - B) &\stackrel{Teo. 1.5(i)}{=} A \cap \overline{(A \cap \overline{B})} \\
&\stackrel{(DM2)}{=} A \cap (\overline{A} \cup \overline{\overline{B}}) \\
&\stackrel{Tab. 1.1(p_4)}{=} (A \cap \overline{A}) \cup (A \cap \overline{\overline{B}}) \\
&\stackrel{Teo. 1.4(ii)}{=} \emptyset \cup (A \cap \overline{\overline{B}}) \\
&\stackrel{Tab. 1.1(p_5)}{=} A \cap \overline{\overline{B}} \\
&\stackrel{Teo. 1.4(iii)}{=} A \cap B
\end{aligned}$$

j.

$$\begin{aligned}
(A - B) - B &\stackrel{Teo. 1.5(i)}{=} (A \cap \overline{B}) \cap \overline{B} \\
&\stackrel{Tab. 1.1(p_3)}{=} A \cap (\overline{B} \cap \overline{B}) \\
&\stackrel{Tab. 1.1(p_1)}{=} A \cap \overline{B} \\
&\stackrel{Teo. 1.5(i)}{=} A - B
\end{aligned}$$

Para prosseguir com esta seção sobre as operações definidas sobre conjuntos será agora apresentada a última operação “clássica”, sendo esta a diferença simétrica.

Definição 1.17 (Diferença simétrica)

Dado dois conjuntos A e B , a diferença simétrica de A e B , denotado por $A \ominus B$, corresponde ao seguinte conjunto:

$$A \ominus B = \{x \mid x \in (A - B) \text{ ou } x \in (B - A)\}$$

Olhando atentamente a definição anterior é fácil notar que o conjunto da diferença simétrica é exatamente a união das possíveis diferenças entre os conjuntos, isto é, a diferença simétrica corresponde a seguinte igualdade: $A \ominus B = (A - B) \cup (B - A)$.

Exemplo 1.19 Seja $A = \{1, 2, 3\}$ e $B = \{3, 4, 5, 2\}$ tem-se que $A \ominus B = \{1, 4, 5\}$.

A seguir será apresentada uma série de importantes resultados com respeito a diferença simétrica.


Teorema 1.8

Sejam A e B subconjuntos quaisquer de um determinado universo \mathbb{U} , tem-se que $A \ominus B = (A \cup B) \cap \overline{(A \cap B)}$.

Demonstração Dado A e B dois subconjuntos quaisquer de um determinado universo \mathbb{U} segue que:

$$\begin{aligned}
A \ominus B &= (A - B) \cup (B - A) \\
&\stackrel{Teo. 1.5(i)}{=} (A \cap \overline{B}) \cup (B \cap \overline{A}) \\
&\stackrel{Tab. 1.1(p_4)}{=} (A \cup (B \cap \overline{A})) \cap (\overline{B} \cup (B \cap \overline{A})) \\
&\stackrel{Tab. 1.1(p_4)}{=} ((A \cup B) \cap (A \cup \overline{A})) \cap ((\overline{B} \cup B) \cap (\overline{B} \cup \overline{A})) \\
&\stackrel{Teo. 1.4(i)}{=} ((A \cup B) \cap \mathbb{U}) \cap (\mathbb{U} \cap (\overline{B} \cup \overline{A})) \\
&\stackrel{Tab. 1.1(p_1, p_5)}{=} (A \cup B) \cap (\overline{B} \cup \overline{A}) \\
&\stackrel{(DM2)}{=} (A \cup B) \cap \overline{(B \cap A)} \\
&\stackrel{Tab. 1.1(p_2)}{=} (A \cap B) \cap \overline{(A \cap B)}
\end{aligned}$$


Corolário 1.1

Sejam A e B subconjuntos quaisquer de um determinado universo \mathbb{U} , tem-se que $A \ominus B = (A \cup B) - (A \cap B)$. 

Demonstração Pelo Teorema 1.8 tem-se que $A \ominus B = (A \cup B) \cap \overline{(A \cap B)}$, mas pelo Teorema 1.5 (i) segue que $(A \cup B) \cap \overline{(A \cap B)} = (A \cup B) - (A \cap B)$, e portanto, $A \ominus B = (A \cup B) - (A \cap B)$. ■

O próximo resultado mostra que a operação de diferença simétrica entre conjunto possui elemento neutro, isto é, existe um conjunto que quando operado com qualquer outro conjunto A , o resultado é o próprio conjunto A .


Teorema 1.9

Para todo A tem-se que $A \ominus \emptyset = A$. 

Demonstração Dado um conjunto A qualquer pelo Corolário 1.1 tem-se que $A \ominus \emptyset = (A \cup \emptyset) - (A \cap \emptyset)$, mas pelas propriedades apresentadas na Tabela 1.1 tem-se: $A \cup \emptyset = A$ e $A \cap \emptyset = \emptyset$. Logo $A \ominus \emptyset = A - \emptyset$, por fim, pelo Teorema 1.6 (a) tem-se que $A - \emptyset = A$, consequentemente, $A \ominus \emptyset = A$. ■

Seguindo com as propriedades que a operação de diferença simétrica possui, o próximo resultado mostra a existência de um elemento que neste manuscrito será chamado de **alternador**, isto é, existe um conjunto que quando operado com qualquer outro conjunto A , o resultado é o complemento deste conjunto A .


Teorema 1.10

Para todo A tem-se que $A \ominus \mathbb{U} = \bar{A}$. 

Demonstração Similar a demonstração do Teorema 1.9, ficando assim como exercício ao leitor. ■

O teorema a seguir mostra que a diferença simétrica entre um conjunto A e seu complementar \bar{A} é exatamente igual a totalidade do universo do discurso em que estes conjuntos estão inseridos.


Teorema 1.11

Para todo A tem-se que $A \ominus \bar{A} = \mathbb{U}$. 

Demonstração Dado um conjunto A qualquer e seu complementar \bar{A} tem-se pelo Corolário 1.1 que $A \ominus \bar{A} = (A \cup \bar{A}) - (A \cap \bar{A})$, mas pelo Teorema 1.4 tem-se que $A \cup \bar{A} = \mathbb{U}$ e $A \cap \bar{A} = \emptyset$, consequentemente, $A \ominus \bar{A} = \mathbb{U} - \emptyset$, mas pelo Teorema 1.6 tem-se que $\mathbb{U} - \emptyset = \mathbb{U}$, e portanto, $A \ominus \bar{A} = \mathbb{U}$. ■

Continuando a estudar a diferença simétrica o próximo teorema mostra que a diferença simétrica entre um conjunto A e ele mesmo é exatamente igual ao conjunto vazio.


Teorema 1.12

Para todo A tem-se que $A \ominus A = \emptyset$. 

Demonstração Dado um conjunto A qualquer tem-se pelo Corolário 1.1 que vale a seguinte igualdade, $A \ominus A = (A \cup A) - (A \cap A)$. Mas pelas propriedades apresentadas na Tabela 1.1 tem-se que $(A \cup A) = (A \cap A) = A$, logo $A \ominus A = A - A$, mas pelo Teorema 1.6 tem-se que $A - A = \emptyset$, portanto, $A \ominus A = \emptyset$. ■

Anteriormente foi mostrado que a diferença entre conjuntos não era comutativa (Exemplo 1.17), o próximo resultado contrasta esse fato com respeito a diferença simétrica.

Teorema 1.13

Para todo A e B tem-se que $A \ominus B = B \ominus A$. 

Demonstração Dado dois conjuntos A e B tem-se pelo Corolário 1.1 que vale a seguinte igualdade, $A \ominus B = (A \cup B) - (A \cap B)$, mas pela propriedade de comutatividade de \cup e de \cap (ver Tabela 1.1) tem-se que $A \cup B = B \cup A$ e $A \cap B = B \cap A$, logo tem-se que $A \ominus B = (B \cup A) - (B \cap A)$, mas pelo Corolário 1.1 tem-se que $(B \cup A) - (B \cap A) = B \ominus A$, e portanto, $A \ominus B = B \ominus A$. ■

Teorema 1.14

Para todo A, B e C tem-se que $(A \ominus B) \ominus C = A \ominus (B \ominus C)$. ♡

Demonstração A prova deste teorema sai direto da definição de diferença simétrica e assim ficará como exercício ao leitor. ■

Teorema 1.15

Para todo A e B tem-se que $\overline{(A \ominus B)} = (A \cap B) \cup (\overline{A} \cap \overline{B})$. ♡

Demonstração Para todo A e B segue que:

$$\begin{aligned}
 \overline{(A \ominus B)} &\stackrel{\text{Teo. 1.8}}{=} \overline{((A \cup B) \cap \overline{(A \cap B)})} \\
 &\stackrel{\text{(DM2)}}{=} \overline{(A \cup B)} \cup \overline{\overline{(A \cap B)}} \\
 &\stackrel{\text{(DM1)}}{=} (\overline{A} \cap \overline{B}) \cup \overline{\overline{(A \cap B)}} \\
 &\stackrel{\text{(DM2)}}{=} (\overline{A} \cap \overline{B}) \cup \overline{(\overline{\overline{A \cap B}})} \\
 &\stackrel{\text{(DM1)}}{=} (\overline{A} \cap \overline{B}) \cup \overline{(\overline{\overline{A}} \cap \overline{\overline{B}})} \\
 &\stackrel{\text{Tab. 1.1}(p_2)}{=} \overline{(\overline{\overline{A}} \cap \overline{\overline{B}})} \cup (\overline{A} \cap \overline{B}) \\
 &\stackrel{\text{Teo. 1.4}(iii)}{=} (A \cap B) \cup (\overline{A} \cap \overline{B})
 \end{aligned}$$

1.3 Operações generalizadas

Agora após a apresentação de todas as operações básicas sobre conjuntos este manuscrito continua apresentando a forma generalizada das operações de união e interseção.

Definição 1.18 (União generalizada)

Dado uma família A então a união generalizada dos conjuntos em A corresponde respectivamente a:

$$A_{\cup} = \bigcup_{x \in A} x$$

Exemplo 1.20 Dado a família $A = \{\{2, 4\}, \{-1, 2\}, \{4, 9, 8, -1\}\}$ tem-se que:

$$A_{\cup} = \{2, 4, -1, 9, 8\}$$

Exemplo 1.21 Seja $A = \{\{a, b\}, \{a\}, \{b\}, \{c\}\}$ tem-se que a união generalizada dos elementos de A corresponde ao conjunto $A_{\cup} = \{a, b, c\}$.

É fácil perceber pela própria definição que a união generalizada só será vazia se todos os membros da família A forem exatamente iguais ao conjunto vazio. De forma dual tem-se a definição generalizada da interseção como se segue.

Definição 1.19 (Interseção generalizada)

Dado uma família A então a interseção generalizada dos conjuntos em A corresponde respectivamente a:

$$A_{\cap} = \bigcap_{x \in A} x$$



Exemplo 1.22 Seja $D = \{\mathbb{Z}_+, \{0, -1, -2, -3\}, (\mathbb{Z}_- \cup \{0\})\}$, a interseção generalizada de D corresponde ao conjunto $D_{\cap} = \{0\}$.

Exemplo 1.23 Dado $A = \{\{a, t, c, g\}, \{v, x, a, g, d\}, \{z, b, a, y, g\}, \{g, b, a\}\}$ tem-se que $A_{\cap} = \{a, g\}$.

Observação: Vale destacar que as igualdades nas Definições 1.18 e 1.19 são sustentadas pelas propriedades da idempotência, associatividade e comutatividade descritas na Tabela 1.1, para mais detalhes consulte [16].

Como dito em [16, 37], quando A é uma família com uma quantidade de n conjuntos, isto é, quanto tem-se que $A = \{x_1, \dots, x_n\}$, é comum reescrever a definição da união e da interseção generalizada usando as seguintes igualdades:

$$A_{\cup} = x_1 \cup \dots \cup x_n$$

e

$$A_{\cap} = x_1 \cap \dots \cap x_n$$

ou ainda:

$$A_{\cup} = \bigcup_{i=1}^n x_i$$

e

$$A_{\cap} = \bigcap_{i=1}^n x_i$$

Teorema 1.16

Se A é uma família, então:

- i. $\overline{A_{\cup}} = \bigcap_{x \in A} \bar{x}$.
- ii. $\overline{A_{\cap}} = \bigcup_{x \in A} \bar{x}$.



Demonstração A prova segue da aplicação das leis De Morgan, e ficará como exercício ao leitor. ■

1.4 Partes e partições

Como já mencionando algumas vezes anteriormente uma família é um conjunto cujo os elementos são também conjuntos. Agora dado um conjunto A qualquer, em algum momento possa ser que seja necessário (por interesse prático ou teórico) trabalhar com a família dos subconjuntos deste conjunto A , note porém, que qualquer elemento desta família é uma parte do conjunto A , ou seja, a família reuni as partes de A , a seguir é definido formalmente o conceito de família das partes obtida a partir de um determinado conjunto.

Definição 1.20 (Conjunto das partes)

Seja A um conjunto. O conjunto das partes^a de A , é denotada por $\wp(A)$, e corresponde a seguinte família de conjuntos:

$$\wp(A) = \{x \mid x \subseteq A\}$$

^aEm alguns livros é usado o termo conjunto potência em vez do termo conjunto das partes, nesse caso é usado a notação 2^A para denotar tal família de conjuntos, por exemplo ver [37].



Uma propriedade interessante do conjuntos das partes como dito em [36], é que se A for da forma $A = \{x_1, \dots, x_n\}$ para algum $n \in \mathbb{N}$, então pode-se mostrar que $\wp(A)$ terá exatamente 2^n elementos.

Exemplo 1.24 Seja $A = \{a, b, c\}$ tem-se que $\wp(A) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$.

Exemplo 1.25 Dado o conjunto $X = \{1\}$ tem-se que $\wp(X) = \{\emptyset, \{1\}\}$.

Exemplo 1.26 Seja $A = \emptyset$ tem-se que $\wp(A) = \{\emptyset\}$.

Além da ideia de conjunto das partes, uma outra família muito importante dentro da teoria dos conjuntos é a família das partições de um conjunto.

Definição 1.21 (Partição)

Seja A um conjunto não vazio, uma partição é uma família não vazia de subconjuntos disjuntos de A , ou seja, uma família $\{x_i \mid x_i \subseteq A\}$ tal que as seguintes condições são satisfeitas:

- (1) Para todo $y \in A$ tem-se que existe um único i tal que $y \in x_i$ para algum $x_i \subseteq A$.
- (2) Para todo i e todo j sempre que $i \neq j$, então $x_i \cap x_j = \emptyset$.



Como dito em [37] os elementos em uma partição são chamados de **células**.

Exemplo 1.27 Dado o conjunto $A = \{0, 1, 2, 3, 4, 5\}$ tem-se que:

- (a) $R = \{\{1, 5\}, \{2, 1, 4\}, \{0, 3\}\}$ não é uma partição de A pois $\{1, 5\} \cap \{2, 1, 4\} = \{1\}$, e portanto, não são disjuntos.
- (b) $S = \{\{1, 5\}, \{0, 4\}, \{3\}\}$ não é uma partição de A pois o elemento $2 \in A$ não pertence a nenhum dos conjuntos em S .
- (c) $T_1 = \{\{0, 5\}, \{1, 3, 4\}, \{2\}\}$ e $T_2 = \{\{0, 1\}, \{4, 5\}, \{3, 2\}\}$ são ambas partições do conjunto A pois satisfazem todas as condições apresentadas na Definição 1.21.

Observação: É claro que uma partição de um conjunto A é vazia se, e somente se, $A = \emptyset$.

Questionário do Capítulo

1. Para cada um dos conjuntos a seguir, determine uma propriedade que define o conjunto e escreva os conjuntos na notação compacta.
 - (a). $\{0, 2, 4, 6, 8, 1, 3, 5, 7, 9\}$.
 - (b). $\{-2, -4, -6, -8, 0, 6, 4, 8, 2\}$.
 - (c). $\{3, 5, 7, 9, 13, 15, 17, \dots\}$.
 - (d). $\{a, c, s\}$
 - (e). $\{2, 3, 5, 7, 11, 13, 17, 19, \dots\}$.
 - (f). $\{1, 4, 9, 16, 25, 36, 64, 81, 100\}$.
 - (g). $\{3, 6, 9, 12, 15, 18, 21, \dots\}$.
 - (h). $\{\frac{1}{2}, \frac{2}{4}, \frac{3}{6}, \frac{4}{8}, \frac{5}{10}, \dots\}$
2. Escreva os seguintes conjuntos em notação compacta.

- (a). Conjunto de todos os países da América do sul.
 - (b). Conjunto de planetas do sistema solar.
 - (c). Conjunto dos números reais maiores que 1 e menores que 2.
 - (d). Conjunto de estados brasileiros cujo nome começa com a letra “R”.
 - (e). Conjunto dos times nordestinos que já foram campeões da primeira divisão do campeonato brasileiro de futebol.
3. Escreva as sentença a seguir de forma apropriada usando a linguagem da teoria ingênua dos conjuntos.
- (a). x não pertence ao conjunto A .
 - (b). -2 não é um número natural.
 - (c). O símbolo π representa um número real.
 - (d). O conjunto das vogais não é subconjunto do conjunto das consoantes.
 - (e). y é um número inteiro, porém não é um número maior que 10.
 - (f). D é o conjunto de todos os múltiplos de -3 que são maiores que 1.
4. Considere o conjunto de letras $K = \{b, t, s\}$ responda falso ou verdadeiro e justifique sua resposta:
- (a). $s \in K$?
 - (b). $t \subset K$?
 - (c). $K \not\subset K$?
 - (d). $\{b\} \in K$?
 - (e). $K - \{a\} = K$?
5. Considere cada conjunto a seguir e escreva todos os seus subconjuntos.
- (a). $B = \{1, 2, 3\}$.
 - (b). $F = \{a, b, c, d\}$.
 - (c). $N = \{\emptyset\}$.
 - (d). $R = \{\emptyset, \{\emptyset\}\}$.
 - (e). $P = \{\{a, b\}, \{c, d\}, \{a, f\}, \{a, b, c\}, \emptyset\}$.
6. Considerando o universo dos números naturais dado os subconjuntos: $A = \{1, 2, 3, 4, 5\}$, $B = \{x \in \mathbb{N} \mid x^2 = 9\}$, $C = \{x \in \mathbb{N} \mid x^2 - 4x + 6 = 0\}$ e $D = \{x \in \mathbb{N} \mid x = 2k \text{ para algum } k \in \mathbb{N}\}$, complemente as frase com os símbolos $\subseteq, \not\subseteq$.
- (a). $A \subseteq B$.
 - (b). $C \subseteq B$.
 - (c). $D \subseteq C$.
 - (d). $B \subseteq A$.
 - (e). $A \subseteq D$.
 - (f). $C \subseteq A$.
 - (g). $D \subseteq B$.
 - (h). $B \subseteq \mathbb{N}$.
 - (i). $\mathbb{N} \subseteq D$.
 - (j). $A \subseteq \mathbb{N}$.
 - (k). $A \subseteq \mathbb{Z}_-$.
 - (l). $\mathbb{N} \subseteq D$.
 - (m). $\mathbb{N} \subseteq C$.
 - (n). $C \subseteq \mathbb{N}$.
 - (o). $\{6\} \subseteq C$

7. Complete as sentença da teoria dos conjuntos com \in , \subseteq e $\not\subseteq$.

- (a). $2 \underline{\hspace{1cm}} \{1, 2, 3\}$.
- (b). $\{2\} \underline{\hspace{1cm}} \{1, 2, 3\}$.
- (c). $\{1\} \underline{\hspace{1cm}} \{\{1\}, \{2\}, \{3\}\}$.
- (d). $\emptyset \underline{\hspace{1cm}} \{1\}$.
- (e). $\emptyset \underline{\hspace{1cm}} \{\emptyset\}$.
- (f). $\{3\} \underline{\hspace{1cm}} \emptyset$.
- (g). $\mathbb{N} \underline{\hspace{1cm}} \{2, 3, 6\}$.
- (h). $\{\{\emptyset\}, \emptyset\} \underline{\hspace{1cm}} \{\{\{\emptyset\}, \emptyset\}, \emptyset\}$.
- (i). $a \underline{\hspace{1cm}} \{\{a\}, b\} \underline{\hspace{1cm}} \{a, b, c\}$.
- (j). $0 \underline{\hspace{1cm}} \mathbb{Z}_+^*$.
- (k). $\frac{1}{0} \underline{\hspace{1cm}} \mathbb{Q}$.
- (l). $\emptyset \underline{\hspace{1cm}} \emptyset$.
- (m). $\{1, 2, 4\} \underline{\hspace{1cm}} \{2, 4, 6\} \underline{\hspace{1cm}} \{y \mid y = 2x \text{ para algum } x \in \mathbb{N}\}$.
- (n). $\{1\} \underline{\hspace{1cm}} \mathbb{R}$.
- (o). $\frac{3}{4} \underline{\hspace{1cm}} \mathbb{N}$.

8. Justifique as seguintes afirmações.

- (a). $\{\frac{2}{x} \mid x - 1 > 0 \text{ com } x \in \mathbb{N}\}$ não é subconjunto de \mathbb{N} .
- (b). $\{2, 3, 4, 6, 8\}$ não é subconjunto de $\{x \in \mathbb{N} \mid x = 2k \text{ para algum } k \in \mathbb{N}\}$.
- (c). $\{1, 2, 3\}$ é um subconjunto próprio do conjunto $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$.
- (d). $\{0, 5\}$ é subconjunto de \mathbb{Z} mas não é subconjunto de \mathbb{Z}^* .
- (e). $\{x \mid x + x = x\}$ é subconjunto próprio de \mathbb{N} .
- (f). Existem exatamente 15 subconjuntos próprios do conjunto $\{2, 3, 5, 7\}$.
- (g). Não existem subconjuntos próprios do conjunto \emptyset .
- (h). Sempre que $A \subset B$ e $A_0 \subset A$, tem-se que A_0 é também um subconjunto próprio de B .
- (i). O conjunto $\{2\}$ tem um único subconjunto próprio.
- (j). O conjunto $\{x \in \mathbb{N} \mid 0 < x < 3\}$ tem exatamente 3 subconjuntos próprios.

9. Considerando o universo $\mathbb{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$ e seus subconjuntos $A = \{2, 4, 6, 8\}$, $B = \{1, 3, 5, 7, 9\}$, $C = \{1, 2, 3, 4, 0\}$ e $D = \{0, 1\}$ exiba os seguintes conjuntos:

- (a). $A \cup B$.
- (b). $C \cup D$.
- (c). $D \cap A$.
- (d). $B \cap C$.
- (e). $A \cap (B \cup D)$.
- (f). $D \cap (A \cap C)$.
- (g). $(A \cap B) \cup (D \cap C)$.
- (h). $(\mathbb{U} \cap A) \cup D$.
- (i). $(D \cup A) \cap C$.
- (j). $D \cap (B \cup A)$.
- (k). $A \cup \overline{B}$.
- (l). $\overline{(C \cap B)} \cup D$.
- (m). $\overline{D \cap A}$.
- (n). $B \cap \overline{C}$.

- (o). $A \cap \overline{(\overline{B} \cup D)}$.
- (p). $D \cap (A \cap C)$.
- (q). $\overline{(A \cap B)} \cup (\overline{D} \cap C)$.
- (r). $(\overline{U} \cap \overline{A}) \cup D$.
- (s). $\overline{(D \cup A)} \cap \overline{C}$.
- (t). $\overline{D} \cap (B \cup A)$.
- (u). $\overline{D - A}$.
- (v). $\overline{(A - B)} \cap \overline{C}$.
- (w). $A \cap (\overline{B} - D)$.
- (x). $D \cap (A - C)$.
- (y). $\overline{C} - D$.
- (z). $D - A$.
10. Considerando o universo $U = \{a, b, c, d, e, f, g, h, i, j\}$ e seus subconjuntos $A = \{b, d, f, h\}$, $B = \{a, c, e, g, i\}$, $C = \{a, b, c, d, j\}$ e $D = \{a, j\}$ exiba os seguintes conjuntos:
- (a). $\overline{B} - C$.
- (b). $A - (B \cup D)$.
- (c). $\overline{(A - (A \cap B)) - ((\overline{D} \cap C) - A)}$.
- (d). $\overline{(U - \overline{C})} - D$.
- (e). $A \oplus (B \cup D)$.
- (f). $\overline{(A \oplus (A \cap B)) \oplus ((\overline{D} \cap C) \oplus A)}$.
- (g). $\overline{(U \oplus \overline{C})} \oplus D$.
- (h). $\overline{D \oplus A}$.
- (i). $(A \oplus B) \cap \overline{C}$.
- (j). $\overline{C} \oplus D$.
- (k). $D \oplus A$.
- (l). $\overline{\overline{B} \oplus C}$.
- (m). $A \cap (\overline{B \oplus D})$.
- (n). $D \cap (A \oplus C)$.
11. Uma aluna do curso de Ciência da Computação realizou uma pesquisa sobre três ritmos (A, B e C) presentes no aplicativo de música *Spotify* com seus colegas de classe para seu trabalho na disciplina de estatística, e levantou os dados expostos na Tabela 1.2.

Total de entrevistados	Ouvem A	Ouvem B	Ouvem C	Ouvem A e B	Ouvem A e C	Ouvem B e C	Ouvem A, B e C	Não ouvem nenhum dos ritmos
23	8	4	6	2	3	1	1	10

Tabela 1.2: Tabela com dados fictício da pesquisa sobre ritmos no *Spotify*.

- (a). Qual é o número de entrevistados que escutam apenas o ritmo A?
- (b). Quantos entrevistados não escutam o ritmo C?
- (c). Qual é o número de entrevistados que escutam algum dos ritmos?
- (d). Quantos entrevistados escutam o ritmo B ou C, mas não escutam o ritmo A?
12. Dado os conjuntos $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$ e $C = \{1, 5, 6\}$ construa um conjunto X com exatamente 4 elementos tal que $A \cap X = \{3\}$, $B \cap X = \{3, 5\}$ e $C \cap X = \{5, 6\}$.
13. Considere o banco de dados representado na Tabela 1.3. Esboce o conjunto gerado por cada *Query* detalhada abaixo e relacione essas *Queries* com as operações sobre conjuntos.

id	Nome	Salário	Idade	Sexo
23	Júlio	2.300,00	34	M
102	Patrícia	4.650,00	23	F
33	Daniel	1.375,00	20	M
43	Renata	6.400,00	24	F
23	Rafaela	1.800,00	19	F
57	Tadeu	14.450,00	54	M

Tabela 1.3: Uma base de dados representada como uma tabela.

- (a). O conjunto dos id's onde o sexo é igual a F e o salário não é inferior a 2.000,00.
- (b). O conjunto dos salários em que a idade não é superior a 35 ou o sexo é igual a M.
- (c). O conjunto de todos os nome em que a idade não é maior que 30 ou id é menor que 65.
14. Exiba os seguintes conjuntos.
- (a). $\varnothing(\{1, 2, 3\})$.
- (b). $\varnothing(\varnothing(\{0, 1\}))$.
- (c). $\varnothing(\{\mathbb{N}\})$.
- (d). $\varnothing(\{1, \{2\}, \{1, \{2\}\})$.
- (e). $\varnothing(\{1, \{1\}, \{2\}, \{3, 4\}\})$.
- (f). $\varnothing(\varnothing(\{1, 2\})) - \varnothing(\{0, 1\})$.
- (g). $\varnothing(\{a, b, c, g\} \ominus \{g, e, f, d\})$.
- (h). $\varnothing(\varnothing(\varnothing(\{0, 1\})) \cup \varnothing(\{1, 2, 3\}))$.
- (i). $\varnothing(\varnothing(\emptyset) - \emptyset)$.
- (j). $\varnothing(\{2, 3, 4\} \cap (\{-1, 3\} \cup \{-5\}))$
15. Considere o universo $\mathbb{U} = \{a, b, c, d, e, f, g\}$ e seus subconjuntos $A = \{d, e, g\}$, $B = \{a, c\}$, $C = \{b, e, g\}$ calcule e exiba os seguintes conjuntos.
- (a). $\varnothing(C)$.
- (b). $\varnothing(A) - \varnothing(\overline{B})$.
- (c). $\varnothing((A \cup B) \ominus C)$.
- (d). $\varnothing((\overline{A} \cup B)) \ominus \varnothing(C)$.
- (e). $\varnothing((\overline{C \cap B}) - (\overline{A \cap C}))$
- (f). $\varnothing(C) - (\varnothing(A) \ominus \varnothing(B))$.
- (g). $\varnothing(\overline{A}) \ominus ((\varnothing(C) \cap \varnothing(B)) - \varnothing(A))$.
- (h). $\varnothing(\varnothing(A)) - \varnothing(\varnothing(B))$.
- (i). $\varnothing(\varnothing(\overline{C})) \ominus \varnothing(\varnothing(B))$.
- (j). $\varnothing(\mathbb{U})$.
16. Dado o conjunto $A = \{a, b, c, d, e, f, g\}$ diga se os conjuntos a seguir são ou não partições de A, justifique todas as suas resposta.
- (a). $P_1 = \{\{a, c, e\}, \{b\}, \{d, g\}\}$.
- (b). $P_2 = \{\{a, g, e\}, \{c, d\}, \{b, e, f\}\}$.
- (c). $P_3 = \{\{a, b, e, g\}, \{c\}, \{d, f\}\}$.
- (d). $P_4 = \{\{a, b, c, d, e, f, g\}\}$.
- (e). $P_5 = \{\{a, b, d, e, g\}, \{f, c\}\}$.
- (f). $P_6 = \{\{a, b, c, d, e\}, \{e, f, g\}\}$.
- (g). $P_7 = \{\{b, c, d, e, f, g\}, \{a\}, \{b, a, c\}\}$.
- (h). $P_8 = \{\{a, b, c, d, e, f, g\}, \{e, d\}\}$.

- (i). $P_9 = \{\{a\}, \{b\}, \{c\}, \{d, e\}, \{f\}, \{g\}\}$.
- (j). $P_{10} = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}\}$
17. Considere o universo $\mathbb{U} = \{a, b, c, d, e, f, g\}$ e seus subconjuntos $A = \{d, e, g\}$, $B = \{a, c\}$, $C = \{b, e, g\}$ exiba duas partições diferentes para cada um dos conjuntos a seguir.
- (a). $C - \bar{A}$.
- (b). $A - \bar{B}$.
- (c). $(A \cup B) \ominus C$.
- (d). $(\bar{A} \cup B) \ominus C$.
- (e). $\overline{(C \cap B)} - (\bar{A} \cap C)$
- (f). $C - (A \ominus B)$.
- (g). $\bar{A} \ominus ((C \cap B) - A)$.
- (h). $A - B$.
- (i). $\bar{C} \ominus B$.
- (j). $\varnothing(\mathbb{U})$.

Capítulo Métodos de Demonstração

Tópicos

❏ Introdução

❏ Demonstrando Implicações

2.1 Introdução

No capítulo anterior, o leitor encontrou diversas demonstrações dentro da teoria intuitiva (ou Cantoriana) dos conjuntos. Para um leitor iniciante, talvez tenha sido um tanto quanto complicado entender a metodologia usada para construir tais demonstrações. E uma vez que, as demonstrações são figuras de interesse central no cotidiano dos matemáticos, cientistas da computação e engenheiros de software, em especial aqueles que trabalham com métodos formais, este texto irá fazer uma breve pausa no estudo da teoria dos conjuntos, para apresentar um pouco de teoria da prova ao leitor.

Este capítulo começa então com o seguinte questionamento: Do ponto de vista da ciência da computação qual a importância das demonstrações? Bem a resposta a essa pergunta pode ser dada de dois pontos de vista, um teórico (purista) e um prático (aplicado ou de engenharia).

Na perspectiva de um cientista da computação puro, as demonstrações de teoremas são a principal ferramenta para investigar os limites dos diferentes modelos de computação propostos [33, 35], assim sendo é de suma importância que o estudante de graduação em ciência da computação receba em sua formação pelo menos o básico para dominar a “arte” de provar teoremas, sendo assim preparado para o estudo e a pesquisa pura em computação e(ou) matemática.

Já na visão prática, só existe uma forma segura de garantir que um *software* está livre de erros, essa “tecnologia” é exatamente a demonstração das propriedades do *software*. É claro que, mostrar que um *software* não possui erros vai exigir que o *software* seja visto através de um certo nível de formalismo e rigor matemático, mas após essa modelagem através de demonstrações pode-se garantir que um *software* não apresentará erros, e assim se algo errado ocorrer foi por fatores externos, tais como defeito no *hardware* por exemplo, e não por falha ou erros com a implementação. Este conceito é o cerne de uma área da engenharia de *software* [50], chamada métodos (ou especificações) formais, sendo essa área o ponto crucial no desenvolvimento de *softwares* para sistemas críticos [58]. Isto já mostra a grande importância de programadores e engenheiros de *software* terem em sua formação as bases para o domínio das técnicas de demonstração.

Nas próximas seções deste manuscrito serão descritas as principais técnicas de demonstração de interesse de matemáticos, cientistas da computação e engenheiros formais de *software*.

Observação: Para o leitor que nunca antes teve contato com a lógica matemática recomenda-se que antes de estudar este capítulo, o leitor faça pelo menos um rápido estudo do Capítulo 3.

Para poder falar sobre métodos de demonstração e poder então descrever como os matemáticos, lógicos e cientistas da computação justificam enunciados usando apenas a argumentação matemática, será necessário fixar algumas nomenclaturas e listar alguns conceitos importantes.

Definição 2.1 (Enunciado)

Um enunciado é qualquer frase declarativa que possa ser expressa na linguagem da lógica simbólica.



Observação: O leitor que conheça lógica nota facilmente que um enunciado é uma proposição ou predicado.

Para que uma demonstração de um enunciado possa ser aceita como correta, é comum exigir que a argumentação de tal demonstração deve conter um alto nível de rigor (formalismo) matemático, além disso, quanto maior for a riqueza¹ de detalhes, mas fácil é de se entender as demonstrações. As ações usadas para realizar demonstrações recebem o nome de métodos (ou estratégias) de demonstrações.

Os métodos de demonstração apresentados neste manuscrito seguem as ideias, a forma e ordem similares as apresentadas em [63]. Para que fique mais simples do leitor iniciante entender, neste manuscrito a tarefa de realizar demonstração acontecerá em duas etapas: (1) será realizada a construção de um rascunho² de prova, e (2) a partir do rascunho gerado no passo anterior será elaborado um texto formal que em sintaxe é a prova em si.

O rascunho é um ambiente na forma de uma tabela subdividida em três colunas, a primeira o marca número de passos (ou estados) da prova, a segunda coluna contém os dados disponíveis e a terceira coluna diz respeito aos objetivos (ou em inglês *Goal*) que devem ser atingidos. Cada linha no rascunho por sua vez é um estado da prova³. Uma prova está terminada (ou completa) quando todos os objetivos tiverem sido eliminados (ou satisfeitos).

Observação: Os objetivos são modificados apenas de duas formas no rascunho, quando é aplicado alguma estratégia de demonstração que altere os objetivos ou quando os mesmo são obtidos na coluna de dados por meio de inferência, neste caso o mesmo deve ser removido da coluna de objetivos.

Nas seções a seguir serão apresentados as diversas estratégias para realizar demonstrações, em cada uma dela será explicado como desenvolver o rascunho de prova de cada estratégia.

2.2 Demonstrando Implicações

Demonstrar uma implicação consiste em construir uma prova (ou argumento) que mostre a veracidade de um enunciado da forma: “Se α , então β ”, com α e β são proposições ou predicados (para detalhes ver a Seção 3.3).

Existem duas estratégias (ou metodologia) distintas para a realizar a demonstração de implicações: a **demonstração direta** e a **demonstração por contra positiva**. Este manuscrito irá primeiro tratar da demonstração direta cuja estratégia está descrita a seguir.

Definição 2.2 (Demonstração Direta)

Dado uma implicação:^a Se α , então β . A estratégia de demonstração direta consiste em:

1. *Supor que o antecedente α é verdadeiro.*
2. *Provar que o consequente β , usando para isso possivelmente o fato de α ser verdadeiro.*

^aEm lógica simbólica seria escrito como $\alpha \Rightarrow \beta$, para detalhes consulte o Capítulo 3.

Observação: De forma objetiva o passo (1) estratégia de demonstração direta de implicações faz com que α se torne um “dado” que poderá ser usado no passo (2) da demonstração.

No que diz respeito ao rascunho antes de aplicar a estratégia de demonstração direta, a prova estará em um estado i com as informações $\gamma_1, \dots, \gamma_n$ estando na coluna de dados e na coluna de objetivos a ser demonstrados

¹A riqueza de detalhes que uma demonstração tem pode variar, a depender para quem a prova se destina, por exemplo, uma prova escrita para um físico em geral não se preocupa com os por menores do linguajar matemático, diferentemente de uma prova escrita para matemáticos, onde os menores detalhes da linguagem matemática são considerados como informações relevantes.

²O leitor deve ter em mente que menos no rascunho é conveniente escrever o máximo das informações na linguagem da lógica de primeira ordem, isso torna mais simples a conversão do rascunho de prova para o texto formal da prova.

³Esse tipo de visão é similar ao que acontece em provador de teoremas como Coq [13] e Lean.

estará exatamente uma implicação $\alpha \Rightarrow \beta$, então ao aplicar o passo (1) da estratégia direta, é criada uma nova linha $i + 1$ no rascunho em que os dados agora são $\gamma_1, \dots, \gamma_n, \alpha$ e o objetivo a ser demonstrado passa a ser β , ou seja, o estado da prova é alterado de forma que α passa a ser um dado e o novo objetivo a ser demonstrado se torna β . Após desenvolver o passo (2), isto é, demonstrar β (talvez usando n linhas) tem-se que β passa a ser um dado no rascunho, ou seja, na linha $i + 1 + n$ tem-se que a coluna de dados é formada por $\gamma_1, \dots, \gamma_n, \alpha, \beta$ e a coluna objetivos estará vazia. A seguir é ilustrado o procedimento de aplicação da demonstração direta com respeito ao rascunho de prova.

Estado	Dados	Objetivos
\vdots	\vdots	\vdots
i	$\gamma_1, \dots, \gamma_n$	$\alpha \Rightarrow \beta$
$i + 1$	$\gamma_1, \dots, \gamma_n, \alpha$	β
\vdots	\vdots	\vdots
$i + 1 + n$	$\gamma_1, \dots, \gamma_n, \alpha, \beta$	

Tabela 2.1: Rascunho genérico ilustrando a aplicação da regra de demonstração direta.

Os dois enunciados apresentados nos Problemas 2.1 e 2.2 a seguir são clássicos resultados sobre os números inteiros, eles serão usado para apresentar ao leitor na prática o uso da estratégia direta de demonstrar implicações, e também irão apresentar o uso do rascunho de prova e a transformação para texto formal de prova matemática.

Problema 2.1 Demonstrar que: Se n é par, então seu quadrado também é par.

Solução Antes de qualquer coisa a implicação será reescrita usando a linguagem matemática adequada, assim tem-se o seguinte enunciado:

$$n = 2i \text{ com } i \in \mathbb{Z} \Rightarrow n^2 = 2j \text{ com } j \in \mathbb{Z}$$

agora pode-se realizar a demonstração usando a estratégia direta.

Estado	Dados	Objetivos
1		$n = 2i \text{ com } i \in \mathbb{Z} \Rightarrow n^2 = 2j \text{ com } j \in \mathbb{Z}$
2	$n = 2i \text{ com } i \in \mathbb{Z}$	$n^2 = 2j \text{ com } j \in \mathbb{Z}$
3	$n = 2i \text{ com } i \in \mathbb{Z}$ e calculando o quadrado tem-se que $n^2 = 2(2i^2)$	$n^2 = 2j \text{ com } j \in \mathbb{Z}$
4	$n = 2i \text{ com } i \in \mathbb{Z}$, fazendo $j = 2i^2$ tem-se que $n^2 = 2j$ com obviamente $j \in \mathbb{Z}$	

Tabela 2.2: Rascunho da demonstração direta do enunciado, Se n é par, então seu quadrado também é par.

Agora pode-se então transformar o rascunho em um texto formal matemático, para isso basta usar as informações contidas na coluna de dados (sem duplicatas), em geral para provar implicações costuma-se iniciar o texto com os termos **assuma**, **suponha** ou **considere que**. Após esses termos iniciais basta seguir apresentando as informações da coluna dados utilizando entre elas texto da linguagem português que melhor se adeque a fazer o casamento entre as informações, a seguir é esboçado um texto obtido a partir do rascunho representado na Tabela 2.2.

Texto da prova: Suponha que $n = 2i$ com $i \in \mathbb{Z}$, assim elevando n ao quadrado tem-se que $n^2 = 2(2i^2)$, agora desde que a multiplicação de dois números inteiros ainda é um número inteiro, tem-se que existirá um $j \in \mathbb{Z}$ tal que $j = 2i^2$ e, portanto, $n^2 = 2j$. Mas desde que $j \in \mathbb{Z}$ tem-se por definição que n^2 é também um número par.

Observação: É importante ressaltar ao leitor que o texto de prova apresentado na solução do Problema 2.1 é apenas um entre muitos textos possíveis para formalizar a escrita da prova.



Nota: O leitor deve ter notado que na solução do Problema 2.1 o termo “Se n é par” do enunciado foi representado na linguagem matemática por $n = 2i$ com $i \in \mathbb{Z}$, a razão disso não é o foco do conteúdo deste manuscrito, além de que, o autor do mesmo considera que o leitor já possui este nível de maturidade matemática. Como esta, diversas outras “traduções” para a linguagem matemática serão feitas de forma direta, sem que exista preocupação por parte do autor em explicá-las.

Problema 2.2 Demonstrar que: Se n é múltiplo de 4, então também é múltiplo de 2.

Solução Como antes a solução se inicia com a reescrita do enunciado usando a linguagem matemática adequada,

$$n = 4i \text{ com } i \in \mathbb{Z} \Rightarrow n = 2j \text{ com } j \in \mathbb{Z}$$

e então é desenvolvido o rascunho de prova a seguir.

Estado	Dados	Objetivos
1		$n = 4i \text{ com } i \in \mathbb{Z} \Rightarrow n = 2j \text{ com } j \in \mathbb{Z}$
2	$n = 4i \text{ com } i \in \mathbb{Z}$	$n = 2j \text{ com } j \in \mathbb{Z}$
3	Mas $4 = 2 \cdot 2$ assim $n = 2 \cdot 2i$	$n = 2j \text{ com } j \in \mathbb{Z}$
4	$n = 2j \text{ com } j = 2i \text{ com } i \in \mathbb{Z}$	

Tabela 2.3: Rascunho da demonstração direta do enunciado, Se n é múltiplo de 4, então também é múltiplo de 2.

O próximo passo como antes é gerar uma “tradução” do rascunho esboçado na Tabela 2.3, ou seja, expressar de forma textual formal o que foi obtido na construção do rascunho de prova é como se segue.

Texto da prova: Suponha que $n = 4i$ com $i \in \mathbb{Z}$, desde que $4 = 2 \cdot 2$ pode-se reescrever a igualdade anterior como sendo $n = 2j$ tal que $j = 2i$ e obviamente $j \in \mathbb{Z}$, agora desde que $n = 2j$ tem-se por definição que n é um múltiplo de 2, o que conclui a prova.

O segundo método para provar implicações é o método da contra positiva (ou contraposição), que como dito em [46], se baseia na equivalência lógica da expressão “Se α , então β ” com a expressão “Se não β , então não α ”. O método de demonstração por contraposição é formalizado a seguir.

Definição 2.3 (Demonstração por Contra positiva)

Dado uma implicação: Se α , então β . A estratégia de demonstração por contra positiva consiste em:

1. Obter a implicação: “Se não β , então não α ”.
2. Demonstrar a implicação “Se não β , então não α ”, usando o método de demonstração direta.



II

LÓGICA

Alice perguntou, “-Gato Cheshire... pode me dizer qual o caminho que eu devo tomar?”

“-Isso depende muito do lugar para onde você quer ir”, disse o Gato.

“-Eu não sei para onde ir!”, respondeu Alice.

“-Se você não sabe para onde ir, qualquer caminho serve”. Falou o gato.

LEWIS CARROLL, ALICE NO PAÍS DAS MARAVILHAS.

Capítulo Introdução à Lógica

Tópicos

- ☐ O que é Lógica?
- ☐ Um Pouco de História
- ☐ Argumentos, Proposições e Predicados
- ☐ Conectivos, Quantificadores e Negação
- ☐ Representação simbólica
- ☐ Lógica e Ciência da Computação
- ☐ Questionário

3.1 O que é Lógica?

Antes de apresentar uma descrição histórica da lógica, este texto começa pela árdua tarefa de apresentar de forma sucinta uma resposta para a pergunta, “**o que é a lógica?**”. Como dito em [9, 20], a palavra lógica e suas derivações são familiares a quase todas (se não todas) as pessoas, de fato, é comum durante o cotidiano do dia a dia as pessoas recorrerem ao uso do termo lógica ou de seus derivados, sendo que na maioria das vezes seu uso está ligada à ideia de obviedade (ou certeza), por exemplo nas frases:

- (a) É lógico que vou na festa.
- (b) É lógico que ciência da computação é um curso difícil.
- (c) Logicamente o Vasco não pode ganhar o título da primeira divisão nacional em 2021.
- (d) Logicamente se eu tomar banho, vou ter que me molhar.

Essa forma de usar os derivados da palavra lógica enquanto entidades para transmissão de certeza pode ser usada como gatilho “fácil e preguiçoso” para enunciar que a lógica se trata de uma ciência (ou disciplina) acerca das certezas sobre os fatos do mundo real.

Existe outra resposta comumente encontrada na literatura acadêmica (ver [1, 9, 42]) para o que seria a lógica, esta segunda alternativa de resposta descreve a lógica como sendo um mecanismo utilizado durante o raciocínio estruturado e correto¹, isto é, uma ferramenta do raciocínio que possibilita a inferência de conclusões a partir de premissas [1, 20, 32], por exemplo, dado as premissas:

- (a) Toda quinta-feira é servido peixe no almoço.
- (b) Hoje é quarta-feira

O raciocínio munido da “ferramenta de inferência” contida na lógica permite deduzir a afirmação: **Amanhã será servido peixe no almoço**, como conclusão. Note que esta segunda resposta estabelece que a lógica é um tipo de procedimento mental capaz de transformar informações de entrada (as premissas) em informações de saída (a conclusão).

Essas duas formas de encarar a lógica não estão totalmente erradas, entretanto, também não exibem de forma completa o real significado do que seria a lógica em si. Uma terceira resposta para a pergunta “O que é a lógica?” aparece na edição de 1953 da Encyclopædia Britannica na seguinte forma: “*Logic is the systematic study of the structure of propositions and of the general conditions of valid inference by a method which abstracts from the content or matter of the propositions and deals only with their logical form*”. Note que essa resposta utiliza-se de autorreferência², pois a mesma tenta definir o que é a lógica em função do termo “forma lógica”.

¹Uma visão semelhante a este é descrita em [39], que diz que a lógica está preocupada com a avaliação de argumentos, com a separação dos argumentos bons dos ruins.

²Autorreferência é um fenômeno que ocorre na língua natural e nas linguagens formais, tal fenômeno consiste de uma oração ou fórmula

Apesar dessa definição recursiva³, a resposta da Encyclopædia Britannica apresenta duas características muito marcantes para a apresentação da lógica enquanto ciência (ou disciplina) nos dias atuais. A primeira característica é a validade das afirmações derivadas (ou concluídas) pelo mecanismos de inferência. A segunda característica é a importância da forma de representação (a escrita) dos termos lógicos.

A validade remonta a ideia de um significado dual (verdadeiro e falso) para as afirmações, ou seja, fornece indícios da existência de interpretações das afirmações, e isto significa que existem diferentes significados para as afirmações a depender de um fator que pode ser chamado de contexto, por exemplo considere a seguinte afirmação:

“O atual presidente americano é um democrata”.

Note que o contexto temporal muda drasticamente o valor lógico interpretativo (semântico) dessa afirmação pois em 2021 essa afirmação pode ser interpretada como verdadeira, porém no ano de 2019 a mesma era falsa. Assim os valores interpretativos (semânticos) dentro do universo da lógica não são imutáveis, isto é, os valores das interpretações da lógica são passíveis de mudança a depender do contexto.

Dado então estes componentes sintáticos e semânticos pode-se concluir a partir das definições linguísticas que a **lógica é uma linguagem**, entretanto, vale salientar que não é uma linguagem natural como o português, como será visto nos próximos capítulos a lógica é uma linguagem formal [12], no sentido de que todas as construções linguísticas possuem uma forma precisa e sem ambiguidade determinada por uma gramática geradora [33, 35], pode-se inclusive estabelecer que a lógica é a linguagem da ciência da inferência racional, ou seja, a linguagem usada para representar argumentos, inferência e conclusões sobre um certo universo do discurso.

3.2 Um Pouco de História

A história do desenvolvimento da lógica remonta até a Grécia antiga e a nomes como: Aristóteles (384-322 a.C.), Sócrates (469-399 a.C.), Zenão de Eléia (490-420 a.C.), Parmênides (515-445 a.C.), Platão (428-347 a.C.), Eudemos de Rodas (350-290 a.C.), Teofrasto de Lesbos (378-287 a.C.), Euclides de Megara (435-365 a.C.) e Eubulides de Mileto⁴ (384-322 a.C.). De fato o nome lógica vem do termo grego *logike*, cunhado por Alexandre de Afrodísias no fim do século II depois de Cristo. Como explicado em [1], os mais antigos registros sobre o estudo da lógica como uma disciplina (ciência) são encontrados exatamente na obra de Aristóteles intitulada como “Γ da metafísica”. Todavia, após seu desenvolvimento inicial dado pelos gregos antigos, a lógica permaneceu quase que intocada⁵ por mais de 1800 anos.

Os primeiros a profanar a santidade da lógica de forma contundente, abalando as estruturas da ideia que a lógica era uma ciência completa⁶ foram os matemáticos George Boole (1815-1864) e Augustus De Morgan (1806-1871), que introduziram a moderna ideia da lógica como uma ciência simbólica, isto é, eles semearam os conceitos iniciais que depois iriam convergir para as ideias da lógica enquanto linguagem formal apresentadas

que refere-se a si mesma de forma direta ou através de alguma sub-frase ou fórmula intermediária, ou ainda por meio de alguma codificação.

³Em matemática a ideia de definição recursiva está ligada a ideia de uma estrutura que apresenta autorreferência.

⁴A quem é creditado o paradoxo do mentiroso.

⁵Aqui não está sendo levando em conta as tentativas de Gottfried Wilhelm Leibniz (1646-1716) de desenvolver uma linguagem universal através da precisão matemática.

⁶No sentido de que não havia nada novo a se fazer, estudar ou provar.

pelo matemático e filósofo alemão Gottlob Frege (1848-1925), que via a lógica como uma linguagem, que continha em seu interior todo o rigor da matemática.

Ainda no século XIX os maiores defensores das ideias de Frege, os britânicos Alfred Whitehead (1861-1947) e Bertrand Russel (1872-1970), usaram muitas de suas ideias e sua linguagem na publicação monumental em três volumes intitulada “*Principia Mathematica*” [5], que é ainda hoje considerada por muitos o maior tratado matemático do século XIX. Como dito em [9], outro influenciado por Frege que apresentou importantes contribuições foi filósofo austríaco Ludwig Wittgenstein (1889-1951), que em seu “*Tractatus Logico-Philosophicus*” apresentou pela primeira vez a lógica proposicional através das tabelas verdade. Muitos autores, como é o caso de [1], consideram que a lógica moderna se iniciou verdadeiramente com a publicação do *Principia*, de fato, alguns usam exatamente a visão de Whitehead que diz: “A lógica atual está para a lógica aristotélica como a matemática moderna está para a aritmética das tribos primitivas”.

Outra vertente emergente na lógica do século XIX era aquela apoiada puramente por interesses matemáticos, isto é, a visão da lógica não apenas como linguagem, mas também como um objeto algebrizável (um cálculo). Tal escola de lógica encontra alguns de seus expoentes nos nomes de: Erns Zermelo⁷ (1871-1953), Thoralf Skolem (1887-1963), Ludwig Fraenkel-Conrad (1910-1999), John von Neumann (1903-1957), Arend Heyting (1898-1980) entre outros. Uma das grandes contribuições feitas por essa escola foi incluir uma formulação explícita e precisa das regras de inferência no desenvolvimentos de sistemas axiomáticos.

Uma ramificação desta escola “matemática” ganhou força na Polônia sobre a tutela e liderança do lógico e filósofo Jan Łukasiewicz (1878-1956), o foco da escola polonesa era como dito em [9], analisar os sistemas axiomáticos da lógica proposicional, lógica modal e das álgebras booleanas. Foi esta escola que primeiro considerou interpretações alternativas da linguagem (da lógica) e questões da meta-lógica tais como: consistência, correte e completude. Por fim, foi na escola polonesa que houve pela primeira vez duas visões separadas sobre a lógica, uma em que a lógica era vista puramente como uma linguagem, e a segunda visão que via a lógica puramente como um cálculo [9].

Instigado pelo problema número dois da lista Hilbert⁸ o jovem matemático e lógico austríaco Kurt Gödel (1906-1978) fez grandes contribuições para a lógica, inicialmente ele provou o teorema da completude para a lógica de primeira ordem em sua tese de doutorado em 1929, tal resultado estabelece que uma fórmula de primeira ordem é dedutível se e somente se ela é universalmente válida [9]. Outra contribuição monumental de Gödel são seus teoremas da incompletude [26], em especial o primeiro que deu uma resposta negativa ao problema número dois da lista Hilbert, de forma sucinta o resultado de Gödel estabelece que não pode haver uma sistematização completa da Aritmética, ou seja, sempre vão existir sentença verdadeiras porém indemonstráveis [1].

Outros contemporâneos de Gödel também contribuíram fortemente para a lógica, Alfred Tarski (1901-1983) foi o responsável pela matematização do conceito de verdade como correspondência [1, 61], já o francês Jacques Herbrand (1908-1931) introduziu as funções recursivas e apresentou os resultados hoje chamados de teoria de Herbrand. Entre os resultados de Herbrand se encontra o teorema que relaciona um conjunto insatisfatível de fórmulas da lógica de primeira ordem com um conjunto insatisfatível de fórmulas proposicionais.

Outra enorme revolução matemática do século XX que foi escrita na linguagem da lógica foi a prova da independência entre a hipótese do *continuum*⁹ e o axioma da escolha da teoria de conjuntos de Zermelo–Fraenkel

⁷Zermelo junto com Fraenkel desenvolveu o sistema formal hoje conhecido como teoria axiomática dos conjuntos.

⁸A lista de Hilbert é um lista inicialmente composta por 10 problemas e depois expandida para 23, que foi apresentada pelo matemático alemão David Hilbert (1862-1943) como uma forma de guia a atenção dos matemáticos no século XX.

⁹A hipótese do *continuum* é uma conjectura proposta por Georg Cantor e que fazia parte da lista inicial de 10 problemas estabelecida

ou teoria dos conjuntos axiomática como também é chamada.

De forma sucinta pode-se então concluir que a lógica uma ciência nascida na Grécia antiga se desenvolveu de forma exponencial após o século XIX, e que seu desenvolvimento foi em boa parte guiado por matemáticos, de fato, pode-se dizer que a lógica contemporânea se caracteriza pela tendência da matematização da lógica [8]. Muitos outros estudiosos, além dos que foram aqui mencionados, também apresentaram resultados diretos em lógica ou em área correlatas como a teoria da prova e a teoria da recursão, tornando a lógica e suas ramificações e aplicações um dos assuntos dominantes no séculos XX e XXI.

3.3 Argumentos, Proposições e Predicados

Como qualquer outra disciplina para entender de fato o que é a lógica deve-se estudar a mesma [20], antes de qualquer coisa é bom saber que diferente de outras ciências, a lógica não apresentar fronteiras bem definidas, na verdade como dito em [42], a lógica pode ser compreendida como a tênue linha que separa as ciências da filosofia e da matemática, no que diz respeito a isto, este manuscrito irá se debruçar primariamente sobre os aspectos matemáticos da lógica.

É sabido que para se estudar uma ciência deve-se saber quais são as entidades fundamentais de interesse dessa ciência, no caso da lógica, estas entidades fundamentais são os argumentos em um discurso. Antes de apresentar a noção formal de argumento é conveniente apresentar a ideia de frase declarativa. As frases declarativas usadas para construção de argumentos são aquelas que como dito em [42], enunciam como as entidades em um certo discurso são ou poderia ter sido, em outras palavras, as frase declarativas falam sobre as propriedades das entidades.

Exemplo 3.1 As frase:

- A lua é feita de queijo.
- O Flamengo é um time carioca.

São ambas frases declarativas. Por outro lado, as frase:

- Que horas são?
- Forneça uma resposta para o exercício.
- Faça exatamente o que eu mandei.
- Cuidado!

Não são frases declarativas.

Uma forma de identificar se uma frase é declarativa é verificado se a mesma admite ser classificada como verdadeira ou falso. Na lógica as frase declarativas podem ser “tipadas” com dois rótulos: **proposições** e **predicados**, formalizados em momentos futuros deste manuscrito.

Definição 3.1 (Argumento)

Um argumento é par formado por dois componentes básicos, a saber:

- (1) *Um conjunto de frases declarativas, em que cada frase é chamada de premissa.*
- (2) *Uma frase declarativa, chamada de conclusão.*



Para representar um argumento pode-se como visto em [20, 42] usar uma organização de linhas, por exemplo, para representar um argumento que possua n premissas primeiro serão distribuídas nas n primeiras

por David Hilbert. Esta conjectura consiste no seguinte enunciado: **Não existe nenhum conjunto com cardinalidade maior que a do conjunto dos números inteiros e menor que a do conjunto dos números reais.**

linhas as tais premissas do argumento depois na linha $n + 1$ é usado o símbolo \therefore para separar as premissas da conclusão, sendo esta última colocada na linha $n + 2$.

Exemplo 3.2 São exemplos de argumentos:

A sopa foi preparada com legumes
Toda quarta-feira é servida sopa para as crianças.
Hoje é quinta-feira.
 \therefore
Ontem as crianças tomaram sopa.

e

A lua é feita de queijo
Os ratos comem queijo
 \therefore
O imperador da lua é um rato.

Definição 3.2 (Proposição)

Uma proposição é uma frase declarativa sobre as propriedades de indivíduos específicos em um discurso. 

Exemplo 3.3 São exemplos de proposições:

- (a) $3 < 5$.
- (b) A lua é feita de queijo.
- (c) Albert Einstein era francês.
- (d) O Brasil é penta campeão de futebol masculino.

Definição 3.3 (Predicados)

Predicados são frase declarativas sobre as propriedades de indivíduos não específicos em um discurso. 

Pela Definição 3.3 pode-se entender que um predicado fala das propriedades de indivíduos sem explicitamente dar nomes a tais indivíduos.

Exemplo 3.4 São exemplos de predicados:

- (a) Para qualquer $x \in \mathbb{N}$ tem-se que $x < x + 1$.
- (b) Para todo $x \in \mathbb{R}$ sempre existem dois números $y_1, y_2 \in \mathbb{R}$ tal que $y_1 < x < y_2$.
- (c) Existe algum professor cujo nome da mãe é Maria de Fátima.
- (d) Há um estado brasileiro que não tem litoral.
- (e) Todo time carioca é Sul-Americano.

Agora note que nas frase (a) e (b) do Exemplo 3.4 o símbolo x se torna um mecanismo que faz o papel dos números naturais e reais respectivamente, mas sem ser os próprios números em si, o mesmo vale para y_1 e y_2 . Similarmente na frase (c) o termo **professor** representa todo um conjunto de pessoas, mas nunca sendo uma pessoa em particular, já na frase (d) o termo **estado brasileiro** representa novamente todos os indivíduos de um conjunto, mas ele nunca é um indivíduo particular. Os termos em um predicado que tem essa capacidade de representação são chamados de **variáveis do predicado**.

Observação: Um predicado que tem suas variáveis substituídas por valores específicos se torna uma proposição.

Exemplo 3.5 Considere o predicado: **Existe algum professor cujo nome da mãe é Maria de Fátima**, se

for atribuído o valor **Valdigleis** no lugar da variável **professor** será gerado a proposição: **O nome da mãe de Valdigleis é Maria de Fátima.**

3.4 Conectivos, Quantificadores e Negação

As proposições e os predicados podem ser classificados em duas categorias: simples ou composto. Uma proposição (ou predicado) é dita(o) composta(o) sempre que for possível dividi-la (o) proposição (predicado) em proposições (predicados) menores. E no caso contrário é dito que a proposição (ou predicado) é simples (ou atômicas).

Definição 3.4 (Conectivos)

Conectivos são termos linguísticos que fazem a ligação entre as proposições ou (e) predicados.

Os principais conectivos são: a conjunção, a disjunção, a implicação e a bi-implicação.

Observação: Dependendo do idioma mais de um termo da linguagem pode representar um determinado conectivo.

A seguir são listados os termos na língua portuguesa que são conectivos, ressaltamos que o símbolo _____ será usado como uma variável para representar a posição de proposições (ou predicados).

Conectivo	Termo em Português
Conjunção	_____ e _____
	_____ mas _____
	_____ também _____
	_____ além disso _____
Disjunção	_____ ou _____
Implicação	Se _____, então _____
	_____ implica _____
	_____ logo, _____
	_____ só se _____
	_____ somente se _____
	_____ segue de _____
	_____ é uma condição suficiente para _____
Bi-implicação	Basta _____ para _____
	_____ é uma condição necessária para _____
	_____ se, e somente se _____
	_____ é condição necessária e suficiente para _____

Tabela 3.1: Termos em português que representamos conectivos.

Exemplo 3.6 Usando as proposições do Exemplo 3.3 e os predicados do Exemplo 3.4 pode-se criar:

- $3 < 5$ e para qualquer $x \in \mathbb{N}$ tem-se que $x < x + 1$.
- Há um estado brasileiro que não tem litoral ou O Brasil é penta campeão de futebol masculino.
- Se para todo $x \in \mathbb{R}$ sempre existem dois números $y_1, y_2 \in \mathbb{R}$ tal que $y_1 < x < y_2$, então Albert Einstein era francês.
- Para qualquer $x \in \mathbb{N}$ tem-se que $x < x + 1$ se, e somente se, para todo $x \in \mathbb{R}$ sempre existem dois números $y_1, y_2 \in \mathbb{R}$ tal que $y_1 < x < y_2$.
- A lua é feita de queijo ou $3 < 5$.

Observação: O Exemplo 3.6 mostra que os conectivos podem ser usado para combinar proposições com proposições, predicados com predicados, predicados com proposições e vice-versa.

Como já mencionado antes um predicado não especifica diretamente os indivíduos, em vez disso, usa variáveis para não mencionar os indivíduos especificamente. Essas variáveis por sua vez, estão conectadas a termos da linguagem que determinam a quantidade de elementos que podem vir a ser atribuído a tais variáveis,

tais termos são chamados de quantificadores. Os quantificadores por sua vez, podem ser tipados em duas categoria: universais e existenciais.

Quando uma variável é ligada a um quantificador universal significa que o predicado será verdadeiro se para a atribuição de cada um dos elementos do universo discurso a proposição gerada com a atribuição é também verdadeira, no caso contrário o predicado é falso. Por outro lado, quando uma variável é ligada a um quantificador existencial significa que tal predicado será verdadeiro se para pelo menos um dos elementos do discurso ao ser atribuído a variável gera uma proposição verdadeira, e no caso contrário o predicado será falso.

De forma similar aos conectivos os quantificadores também são “representados” por termos da língua portuguesa como mostrado na tabela a seguir.

Quantificador	Termo em Português
Universal	Para todo(a) _____
	Para qualquer _____
	Para cada _____
Existencial	Existe _____
	Para algum _____
	Para um _____

Tabela 3.2: Termos em português que representamos quantificadores.

Anteriormente já foi dito que na lógica as proposições e predicados podem ser interpretados como sendo verdadeiros ou falsos, dito isto, para qualquer proposição ou predicado sempre é possível obter uma proposição ou predicado com um valor de interpretação oposta, isto é, se a proposição (ou predicado) original for verdadeira a proposição (ou predicado) oposta será falsa, ou vice-versa. Esse operador que gerar as proposições (ou predicados) opostas(os) é chamado de negação e a tabela a seguir exhibe como os termos na língua portuguesa podem ser usados para representar a negação.

Termos em português
Não _____
É falso que _____
Não é verdade que _____

Tabela 3.3: Termos em português para designar a negação de uma proposição ou predicado.

3.5 Representação simbólica

3.6 Lógica e Ciência da Computação

Para finalizar este capítulo introdutório é conveniente falar mesmo que de forma superficial sobre os tipos de lógica. A lógica assim como a física pode ser dividida em duas categorias ou tipos bem definidos, a saber, clássicas e as não clássicas. Como mencionado em [4, 9], as lógicas clássicas são aquelas que apresentam a característica de obedecer os seguintes princípios:

- **Princípio da não contradição:** Qualquer proposição (ou predicado) não pode ser verdadeira(o) e falsa(o) ao mesmo tempo;
- **Princípio do terceiro excluído:** Toda(o) proposição (ou predicado) só pode ser falsa(o) ou verdadeira(o), não existe uma terceira possibilidade.

Logo a lógica clássica é bi-valorada [4], ou seja, as interpretações sobre as proposições e predicados só podem ser valoradas por dois valores, a saber: verdadeiro ou falso. E por sua vez, a própria lógica clássica é sub-dividida em duas partes, sendo estas: a lógica proposicional e a lógica de primeira ordem (ou lógica dos predicados).

Como respeito a aplicações a lógica clássica tem um papel fundamental e central para a Ciência da Computação, uma vez que, todos computadores são construídos pela combinação de circuitos digitais e estes por sua vez implementam operações da lógica proposicional [1, 59]. Outra área de destaque da aplicação da lógica dentro da Ciência da Computação é no campo de Inteligência Artificial, onde a mesma é o principal formalismo de representação do conhecimento e portanto é muito útil no desenvolvimento de sistemas especialistas e sistemas multi-agentes [9], algumas outras áreas de aplicação da lógica clássica são:

- Banco de dados: através descrição de consultas e no relacionamento das tabelas em bancos de dados dedutivos.
- Ontologias web: como uma linguagem para descrever ontologias e representar o conhecimento.
- Engenharia de software: usada como formalismo para especificação e verificação formal das propriedades dos sistemas¹⁰.

As lógica não clássicas por sua vez, podem se apresentar de duas forma. (1) não obedecem algum dos princípios apresentados acima ou (2) estendem a lógica clássica através de teoremas e meta-teoremas (discutidos nos próximos capítulos) não válidos para as lógicas clássicas. Como exemplos de lógica não clássicas estão: lógica intuicionista [38], lógica paraconsistente [56], lógicas multivaloradas [9, 39], lógicas modais [39] e lógicas temporais [29, 30, 40]. Com respeito a aplicações relacionadas Ciência da Computação tem-se por exemplo:

- A utilização da lógica modal para a verificação da propriedades de sistemas e software [30].
- A lógica temporal usada para especificação e verificação de programas concorrentes [40] e também para especificar circuitos síncronos [29].
- As lógicas multi-valoradas usadas para lidar com a simulação e representação de incertezas presente no raciocínio aproximado [9], principalmente na área de reconhecimento de padrões.

Obviamente com dito em [9], existem muitas outras lógicas não clássicas que têm aplicações ou ainda servem de fundamentação para diversas áreas ou disciplinas da computação. Porém, como esta parte do texto é apenas uma introdução não cabe neste escopo se debriçar tão profundamente assim neste assunto, em capítulos futuros as lógicas não clássicas (em especial a modal) serão estudadas mais a fundo.

Questionário do Capítulo

1. Examine cada uma das frases declarativas abaixo e diga se as mesmas são proposições ou predicados e também diga se são simples ou compostas, justifique suas respostas no caso de proposição (ou predicado) composta(o).
 - (a). Existe um gato amarelo.
 - (b). Alguns patos são marrons.
 - (c). Não é verdade que o gato de Júlio é amarelo.
 - (d). O Flamengo joga hoje.
 - (e). Todos os ratos tem olhos azuis.
 - (f). 4 é o menor número composto pelo produto de primos.
 - (g). Meu cachorro é branco, alguns outros são vermelhos.
 - (h). Alguns gatos são cinza, mas meu gato não é cinza, além disso, Tadeu tem um gato preto ou Sormany tem um rato amarelo.

¹⁰Em especial sistemas criticos como software para controle aéreo são exemplo de sistemas cujas propriedades deve ser especificadas e verificadas com alta precisão matemática dado a importância do mesmo para a manutenção da vida humanas que dependem dele.

- (i). Os carros são amarelos se, e somente, se eles não são italianos.
 - (j). Se todos os jogadores da seleção jogam na Europa ou Neymar está machucado, então o Brasil não vence a Argentina.
 - (k). Basta mais um ponto na carteira de motorista para Sormany perde a aposta ou Juca terá que pagar o almoço.
 - (l). Se Lucas é irmão de Pedro, então Natalia vai casar com Gabriel se, e somente se, Francisco voltar da Espanha.
 - (m). Se $\frac{10^2}{50} = 2!$, então o $\pi - 2x = 0$ para todo $x \in \mathbb{C}$.
 - (n). Se a terra é plana e existe chip nas vacinas, então o Brasil vai conquistar o país de Juvenal.
 - (o). A bola é preta.
 - (p). Eu tirei foto com meu avô hoje.
 - (q). $3 < 5$ segue do fato que o voto no papel é mais rápido que voto eletrônico.
 - (r). O sorvete ser de uva segue do fato de Juliane está grávida.
 - (s). Bill escreveu o DOS em 1978 é condição necessária de Steven ter lançado o *Apple 2*.
 - (t). Se o Cruzeiro é um time da primeira divisão, então todo macaco come macarrão ou não é o caso de Patricia ser professora de matemática.
2. Determine as frases simples (ou atômicas) que compõem as proposições e predicados que se seguem.
- (a). Juca não irá a festa, mas Pedro irá ou Flaviana irá.
 - (b). Fui com a minha família ontem ao parque, e me diverti muito.
 - (c). Juca vai com família ou irá sozinho, mas se Anabel aparecer no parque, então Juca e Paula não vão se diverti.
 - (d). Se Paulo chegou, então ele está na sala. Mas não é verdade que Paulo chegou.
 - (e). Valdi toca clarinete somente se, Katia tocar flauta ou Shizue sabe desenhar com carvão.
 - (f). Eu sou aluno da computação somente se, eu passei em Matemática discreta. Mas eu não passei em Introdução à programação ou reprovei todas as disciplinas do primeiro período.
 - (g). Para qualquer navio no porto, existe um marinheiro bêbado no bar ou todos os soldados estão dormindo na praia.
 - (h). Se Romero tivesse vindo vê o filme, então Katia teria ido para a sorveteria com ele. Mas Romero foi para a praia com Julinha.
 - (i). Todos os números primos são números ímpares ou o número π é o pode ser escrito como produto de números primos.
 - (j). Vou a padaria se, e somente se, estiver fazendo frio ou se Juliane quiser tomar sopa.
 - (k). Não é verdade que a terra é esférica se, e somente se, não existe pessoas morando em Recife.
 - (l). Shizue e Valdi foram para o Chile, mas Luiza também foi.
 - (m). Basta que eu tire 8 em Matemática discreta para eu ter um noite de festa.
 - (n). A gripe é um condição suficiente para ser declarado morto.
 - (o). Se para todo flor existe um vaso, então os jardins de Jaçanã são cheio de cerejeiras.
 - (p). Se ontem choveu a noite toda e hoje é meu aniversário de 14 anos, então Pedro vai a Califórnia se, e somente se, Tom e Frajola forem amigos do Manda-chuva.
 - (q). Não é verdade que Shizue sabe desenhar com carvão, mas sabe desenhar com lápis e pincel.
 - (r). O Catatau comeu o mel somente se, o Puffy saiu para passear com o Jack ou Jerry é vizinho do Mickey.
 - (s). Para todo homem existe uma mulher que é sua mãe ou Valdi toca clarinete.

- (t). Existe um carro amarelo se, e somente se, todas bicicletas são roxas, mas as motos são azuis.

Capítulo Lógica Proposicional

III

ÁLGEBRA UNIVERSAL

Alice perguntou, “-Gato Cheshire... pode me dizer qual o caminho que eu devo tomar?”

“-Isso depende muito do lugar para onde você quer ir”, disse o Gato.

“-Eu não sei para onde ir!”, respondeu Alice.

“-Se você não sabe para onde ir, qualquer caminho serve”. Falou o gato.

LEWIS CARROLL, ALICE NO PAÍS DAS MARAVILHAS.

IV

TEORIA DOS NÚMEROS E COMBINATÓRIA

Alice perguntou, “-Gato Cheshire... pode me dizer qual o caminho que eu devo tomar?”

“-Isso depende muito do lugar para onde você quer ir”, disse o Gato.

“-Eu não sei para onde ir!”, respondeu Alice.

“-Se você não sabe para onde ir, qualquer caminho serve”. Falou o gato.

LEWIS CARROLL, ALICE NO PAÍS DAS MARAVILHAS.

V

LINGUAGENS FORMAIS, AUTÔMATOS E COMPUTABILIDADE

Alice perguntou, “-Gato Cheshire... pode me dizer qual o caminho que eu devo tomar?”

“-Isso depende muito do lugar para onde você quer ir”, disse o Gato.

“-Eu não sei para onde ir!”, respondeu Alice.

“-Se você não sabe para onde ir, qualquer caminho serve”. Falou o gato.

LEWIS CARROLL, ALICE NO PAÍS DAS MARAVILHAS.

Capítulo Introdução

Tópicos

- ☐ Sobre Linguagens Formais
- ☐ Sobre Autômatos Finitos
- ☐ Conceitos Básicos

- ☐ Sobre Gramáticas Formais
- ☐ Questionário

5.1 Sobre as Linguagens Formais

5.2 Sobre Autômatos Finitos

Como dito em [21, 22] 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 memórias¹, estas sendo subdivididas em células, cada uma das quais capaz de comportar um único símbolo por vez.
- Uma unidade de controle² que administra o estado atual do autômato e é responsável por executar as instruções (programa) da máquina.

Com respeito as memórias é comum assumir a existência de um **dispositivo de leitura e (ou) escrita**³ 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 [12].

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 [21, 22], 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).

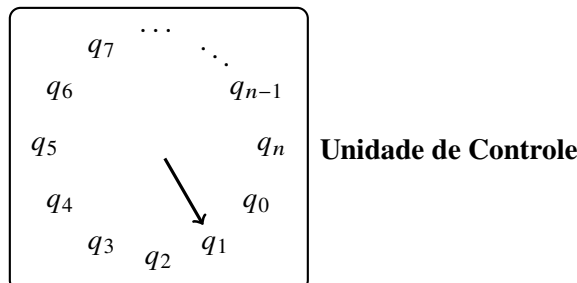


Figura 5.1: Conceito informal de autômato finito com uma única memória retirado de [21].

¹Na literatura também é usado o termo fita em vez de memória [45].

²Também chamada de unidade central de processamento (UCP).

³Também é usado a nomenclatura cabeçote [21, 22].

Formalmente pode-se dizer como apontado em [21], 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 [43], Kleene [34], Mealy [44], Moore [47], Rabin e Scott [51, 52]. De forma geral os autômatos finitos são os mais simples modelos abstratos de máquinas de computação [25], sendo eles máquinas de Turing limitadas.

5.3 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 5.1 (Alfabetos e Palavras)

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 .

Exemplo 5.1 Os conjuntos $\{0, 1, 2, 3\}$, $\{a, b, c\}$, $\{\heartsuit, \spadesuit, \diamondsuit, \clubsuit\}$ e $\{n \in \mathbb{N} \mid n \leq 25\}$ são todos alfabetos, os conjuntos \mathbb{N} e \mathbb{R} não são alfabetos.

Exemplo 5.2 Dado o alfabeto $\Sigma = \{0, 1, 2, 3\}$ tem-se que as sequências 0123, 102345, 1 e 0000 são todas palavras sobre Σ .

Definição 5.2 (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 .

^aPor conta desta notação em alguns texto é usado o termo módulo em vez de comprimento.

Exemplo 5.3 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 $|ddaacc| = 6$.

Observação: Em especial quando $|w| = 1$, é dito que w é uma palavra unitária, isto é, a mesma contém apenas um único símbolo do alfabeto.

Como muito bem explicado em [12, 33, 35], pode-se definir uma série de operações sobre palavras, sendo a primeira delas a noção de concatenação.

Definição 5.3 (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, $w_1 w_2 = a_1 \cdots a_m b_1 \cdots b_n$.

Observação: O leitor deve ficar atento ao fato de que a concatenação apenas combina duas palavras em uma nova palavra, sendo que, não a qualquer tipo de exigência sobre os alfabeto sobre os quais as palavras usadas na concatenação estão definidas.

Exemplo 5.4 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 5.4 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 5.1 (Associatividade da Concatenação)

Para quaisquer w_1, w_2 e w_3 tem-se que $(w_1w_2)w_3 = w_1(w_2w_3)$.



Demonstração 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,

$$\begin{aligned} (w_1w_2)w_3 &= (a_1 \cdots a_i b_1 \cdots b_j) c_1 \cdots c_k \\ &= a_1 \cdots a_i b_1 \cdots b_j c_1 \cdots c_k \\ &= a_1 \cdots a_i (b_1 \cdots b_j c_1 \cdots c_k) \\ &= w_1(w_2w_3) \end{aligned}$$

o que conclui a prova. ■

Sobre qualquer alfabeto Σ sempre é definida uma palavra especial chamada **palavra vazia** [33, 35], esta palavra especial não possui nenhum símbolo, e em geral é usado o símbolo λ para denotar a palavra vazia [12, 22]. Como mencionado em [12, 21] sobre a palavra vazia é importante destacar que:

$$w\lambda = \lambda w = w \quad (5.1)$$

$$|\lambda| = 0 \quad (5.2)$$

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

Definição 5.4 (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:

$$w^0 = \lambda \quad (5.3)$$

$$w^{n+1} = ww^n \quad (5.4)$$



Exemplo 5.5 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 5.6 Seja $u = 01$ e $v = 231$ tem-se que:

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

e também

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

Proposição 5.1

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

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

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



Demonstração Direto das Definições 5.3 e 5.4, e portanto, ficará como exercício ao leitor. ■

Definição 5.5 (Palavra Inversa)

[22] 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 5.7 Dado as palavras $u = aba$, $v = 011101$ e $w = 3021$ tem-se que $u^r = aba$, $v^r = 101110$ e $w^r = 1203$.

Observação: Com respeito a noção de palavra inversa tem-se em particular que vale a seguinte igualdade $\lambda^r = \lambda$.

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 (ver Capítulo 1) 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 5.6 (Potência de um alfabeto)

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

$$\Sigma^0 = \{\lambda\} \quad (5.5)$$

$$\Sigma^{n+1} = \{aw \mid a \in \Sigma, w \in \Sigma^n\} \quad (5.6)$$



Exemplo 5.8 Dado $\Sigma = \{a, b\}$ tem-se que $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$ e $\Sigma^1 = \{a, b\}$

Exemplo 5.9 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 finito⁴.

Definição 5.7 (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:

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i \quad (5.7)$$

e

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i \quad (5.8)$$



Obviamente como dito em [12], o fecho de 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 com destacado em [21] o mesmo corresponde ao monoide livremente⁵ gerado pelo conjunto Σ munida da operação de concatenação.

Definição 5.8 (Prefixos e Sufixos)

Uma palavra $u \in \Sigma^*$ é um prefixo de outra palavra $w \in \Sigma^*$, denotado por $u \leq_p w$, sempre que $w = uv$, com $v \in \Sigma^*$. Por outro lado, uma palavra u é um sufixo de outra palavra w , denotado por $u \leq_s w$, sempre que $w = vu$.



⁴Essa afirmação é facilmente verificável, uma vez que, a mesma nada mais é do que um exemplo de arranjo com repetição.

⁵Relembre que uma álgebra é livremente gerada quando todo elemento possui fatoração única (a menos de isomorfismo).

Exemplo 5.10 Seja $w = abracadabra$ tem-se que 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 5.9 (Conjunto dos Prefixos e Sufixos)

Seja $w \in \Sigma^*$ o conjunto de todos os prefixos de w corresponde ao conjunto:

$$PRE(w) = \{w' \in \Sigma^* \mid w' \leq_p w\} \quad (5.9)$$

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

$$SUF(w) = \{w' \in \Sigma^* \mid w' \leq_s w\} \quad (5.10)$$

Exemplo 5.11 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 5.12 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 5.2

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

- (i) $\#PRE(w) = |w| + 1$.
- (ii) $\#PRE(w) = \#SUF(w)$.
- (iii) $\#(PRE(w) \cap SUF(w)) \geq 1$.

Demonstração 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 $a_1 \cdots a_i \leq_p w$, 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) É suficiente mostrar que $\#SUF(w) = |w| + 1$, para isso como antes sem perda de generalidade assuma que $w = a_1 \cdots a_n$ e assim tem-se que $w \in \Sigma^n$ logo $|w| = n$ com $n \in \mathbb{N}$, dessa forma existem exatamente n palavras da forma $a_i \cdots a_n$ com $1 \leq i \leq n$ tal que $a_i \cdots a_n \leq_s w$, portanto, para todo $1 \leq i \leq n$ tem-se que $a_i \cdots a_n \in SUF(w)$, além disso, é claro que $w = w\lambda$, logo $\lambda \in SUF(w)$, consequentemente, $\#SUF(w) = n + 1 = |w| + 1$, e portanto, $\#PRE(w) = \#SUF(w)$. O caso $w = \lambda$ é trivial e não será demonstrado aqui.
- (iii) Trivial, pois basta notar que $\lambda \in (PRE(w) \cap SUF(w))$, e portanto, tem-se claramente que $\#(PRE(w) \cap SUF(w)) \geq 1$.

Corolário 5.1

Toda palavra tem pelo menos um prefixo e um sufixo.

Demonstração Direto do item (iii) do Teorema 5.2. ■

Seguindo com o texto deste manuscrito pode-se finalmente formalizar o pilar fundamental (a ideia de linguagem) necessário para desenvolver o estudo da computabilidade neste e nos próximos capítulos.

Definição 5.10 (Linguagem)

Dado um alfabeto Σ , qualquer subconjunto $L \subseteq \Sigma^$ será chamado de linguagem.*

Exemplo 5.13 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 [2, 36, 37], isto é, estão definidas sobre as linguagens as operações de união, interseção, complemento, diferença e diferença simétrica. E como para os alfabetos novas operações são definidas.

Definição 5.11 (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:

$$L_1 L_2 = \{xy \in (\Sigma_1 \cup \Sigma_2)^* \mid x \in L_1, y \in L_2\} \quad (5.11)$$

Exemplo 5.14 Dado as três linguagens $L_1 = \{\lambda, ab, bba\}$, $L_2 = \{0^{2n}1 \mid n \in \mathbb{N}\}$ e $L_3 = \{a^p \mid p \text{ é primo}\}$ 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) $L_3 L_1 = \{w \mid w = a^p \text{ ou } w = a^{p+1}b \text{ ou } a^p bba \text{ onde } p \text{ é um número primo}\}$.
- (c) $L_2 L_3 = \{0^{2n}1a^p \mid n \in \mathbb{N}, p \text{ é um número primo}\}$.

Definição 5.12 (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 5.15 Considerando as linguagens L_1, L_2 e L_3 do Exemplo 5.14 e a linguagem $\{01, 011\}$ tem-se que:

- (a) $L_1^r = \{\lambda, ba, abb\}$.
- (b) $L_2^r = \{10^{2n} \mid n \in \mathbb{N}\}$.
- (c) $L_3^r = \{a^p \mid n \in \mathbb{N}, p \text{ é um número primo}\}$.
- (d) $\{01, 011\}^r = \{10, 110\}$.

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 5.13 (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:

$$L^0 = \{\lambda\} \quad (5.12)$$

$$L^{n+1} = LL^n \quad (5.13)$$



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

Definição 5.14 (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.

$$L^+ = \bigcup_{i=1}^{\infty} L^i \quad (5.14)$$

$$L^* = \bigcup_{i=0}^{\infty} L^i \quad (5.15)$$



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

Definição 5.15 (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:

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

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



Exemplo 5.16 Considere a linguagem $\{0, 10, 11010\}$ tem-se que:

$$PRE(\{0, 10, 11010\}) = \{\lambda, 0, 1, 10, 11, 110, 1101, 11010\}$$

$$SUF(\{0, 10, 11010\}) = \{\lambda, 0, 10, 010, 1010, 11010\}$$

Nos próximos capítulos deste manuscrito irão ser apresentadas as formalizações da ideia de linguagens formais na visão “mecânica” de Turing [62], entretanto, em vez de apresentar de forma direta os conceitos ligados as máquinas de Turing e as computações por elas realizadas, este manuscrito opta por fazer um estudo seguindo a ideia dos livros texto de linguagens formais [12, 35, 45], assim sendo, esta parte do manuscrito irá apresentar as linguagens formais da mais simples para a mais complexas seguindo a hierarquia de Chomsky [19], ou seja, serão aqui estudadas as linguagens formais na seguintes ordem: regulares, livres do contexto, recursivas e recursivamente enumeráveis.

5.4 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 [6, 14] pode notar que gramáticas formais são na verdade outro nome para sistemas de reescrita [7].

Definição 5.16 (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 $w \triangleright w'$ onde $w \in (V \cup \Sigma)^+$ e $w' \in (V \cup \Sigma)^*$.

^aTamém é comum encontrar na literatura a nomenclatura regras de produção [12, 35].



Exemplo 5.17 A estrutura $G = \langle \{A, B\}, \{a\}, A, P \rangle$ em que P é formado pelas regras $A \triangleright aABa$, $A \triangleright B$ e $B \triangleright \lambda$ 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 5.17 (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 $XwY \gg_G Xw'Y$, sempre que existe uma regra forma $w \triangleright w' \in P$.



Exemplo 5.18 Dado a gramática do Exemplo 5.17 tem-se que $aABa \gg_G aaABaBa$, pois existe em P a regra $A \triangleright aABa$.

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

Exemplo 5.19 Considerando ainda a gramática exibida no Exemplo 5.17 tem-se que $aABa \gg^* aaaABaBaBa$, uma vez que, $aABa \gg aaABaBa \gg aaaABaBaBa$.

Exemplo 5.20 A estrutura $G = \langle \{A, B, S\}, \{0, 1\}, S, P \rangle$ em que P é formado pelas regras $S \triangleright 11A$, $A \triangleright B0$ e $B \triangleright 000$ é uma gramática formal e assim $11A \gg^* 110000$, pois tem-se que, $11A \gg^* 11B0 \gg^* 110000$.

Como dito em [12], dado uma gramática formal G tem-se que sempre houver uma sequencia de derivações $w_1 \gg w_2 \gg \dots \gg w_n$ acontecer, as palavras w_1, w_2, \dots, 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 5.18 (Igualdade de Derivações)

Dado uma gramática $G = \langle V, \Sigma, S, P \rangle$ e duas derivações $S \gg w_1 \gg^* w_n$ e $S \gg w'_1 \gg^* w'_n$ sobre G , será dito que estas derivações são iguais sempre que $w_i = w'_i$ para todo $1 \leq i \leq n$.



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

Definição 5.19 (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, $\mathcal{L}(G) = \{w \in \Sigma^* \mid S \gg^* w\}$.



Exemplo 5.21 Não é difícil verificar que a gramática do Exemplo 5.17 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.

Questionário do Capítulo

1. Dado o alfabeto $\Sigma = \{a, b, c\}$ e as palavras $u = abcab$, $v = bbccabac$ e $w = ccbabbaaca$ determine:
 - (a). A palavra uv^r .
 - (b). A palavra $(w^r)^2u$.
 - (c). A palavra $((u^r)^2v^0)^r v$.
 - (d). A palavra $uu^2v^r w$.
 - (e). A palavra $((wuv)^r)^2u$.
 - (f). O valor da expressão $|w^3| + 2|v^2u| - |u|$.
 - (g). O valor da expressão $2|w^r| - |uv|$.
 - (h). O valor da expressão $|w^r aaw| - |w|$.
 - (i). O valor da expressão $|uv^r| - 4$.
 - (j). O valor da expressão $\frac{|(w^r)^2u|}{2} - \frac{|u|}{6}$.
2. Demonstre para quaisquer palavras u e v e para todo $n \in \mathbb{N}$ as asserções a seguir.
 - (a). Se u é um prefixo de v , então $|u| \leq |v|$.
 - (b). $|u^n| = n|u|$.
 - (c). $|(uv)^r| = |vu|$.
 - (d). Se $|u| = n$, então $n \leq |uv|$.
3. 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 $|w| = \sqcup\{|w'| \mid w' \in L^3\}$.
4. Prove que para qualquer linguagem L e quaisquer $m, n \in \mathbb{N}$ as seguintes asserções.
 - (a). $(L^m)^n = L^{mn}$.
 - (b). $L^m L^n = L^{m+n}$.
 - (c). $(L^r)^n = (L^n)^r$.
 - (d). $\overline{L^r} = \overline{L^r}$.
 - (e). $PRE(L) = (SUF(L^r))^r$.
5. Dado duas linguagens quaisquer L_1 e L_2 demonstre que:
 - (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 $LL_1 \subseteq LL_2$.
6. Demonstre ou refute o predicado $(\forall L \subseteq \Sigma^*)[(\forall n \in \mathbb{N})[\overline{L}^n = \overline{L^n}]]$.
7. Esboce formalmente em que condições a igualdade $PRE(L) = (SUF(L))^r$ é verdadeira.

Capítulo Linguagens Regulares

Tópicos

- ☐ Autômatos Finitos Determinísticos
- ☐ Autômatos Finitos Não-determinísticos
- ☐ λ -Autômatos Finitos Não-determinísticos
- ☐ Teorema Myhill-Nerode e a minimização
- ☐ Máquinas de Mealy e Moore
- ☐ Notação Matricial
- ☐ Expressões Regulares
- ☐ Gramáticas Regulares
- ☐ Álgebra das Linguagens Regulares
- ☐ Questionário

6.1 Autômato Finito Determinístico

Como explicado em diversas obras tais como [12, 33, 35, 45], os autômatos finitos podem ser separados em dois tipos bem definidos, a saber Autômato Finito Determinístico (AFD) e Autômato Finito Não-determinístico (AFN). Agora este manuscrito inicia o estudo dos AFD apresentando sua forma algébrica equacional.

Definição 6.1 (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, $\delta : Q \times \Sigma \rightarrow Q$ 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.

^aEm algumas referências também é usado o termo conjunto de estados de aceitação [23].

Exemplo 6.1 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 6.2 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:

$$\begin{aligned} \delta(q_0, a) &= q_1 & \delta(q_1, a) &= q_2 & \delta(q_2, a) &= q_1 \\ \delta(q_0, b) &= q_1 & \delta(q_1, b) &= q_2 \end{aligned}$$

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.



Nota: Para simplificar a escrita neste manuscrito as siglas AFD e AFN serão usado tanto para designar o singular quanto o plural, ficando a distinção a critério dos conectivos antes de cada sigla.

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 [21]. 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 6.3 A representação por grafo de transição do AFD descrito no Exemplo 6.1 corresponde a figura a seguir.

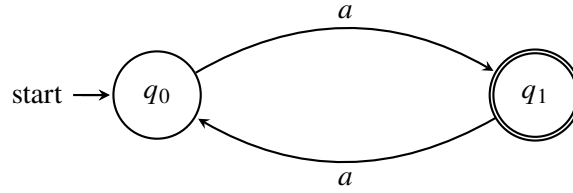


Figura 6.1: Representação visual do AFD no Exemplo 6.1.

Exemplo 6.4 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 6.2 a seguir.

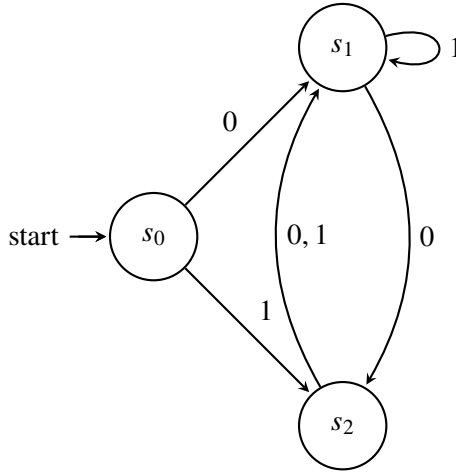


Figura 6.2: Representação visual do AFD S do Exemplo 6.4.

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 6.2 (Função de Transição Estendida)

Seja $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ um AFD a função δ é estendida para uma função $\widehat{\delta} : Q \times \Sigma^* \rightarrow Q$ usando recursividade como se segue.

$$\widehat{\delta}(q, \lambda) = q \quad (6.1)$$

$$\widehat{\delta}(q, wa) = \delta(\widehat{\delta}(q, w), a) \quad (6.2)$$

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 6.3 (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 6.5 Considere o AFD do Exemplo 6.1 e a palavra de entrada $aaaa$ tem-se que a computação desta palavra corresponde a:

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

Exemplo 6.6 Considere o AFD do Exemplo 6.4 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 posse da definição de computação pode-se formalizar o conceito de reconhecimento (ou aceitação) de palavras nos AFD.

Definição 6.4 (Reconhecimento de palavras em AFD)

[12] 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 $\delta(q_0, a_1) = q_{i_1}, \delta(q_{i_1}, a_2) = q_{i_2}, \dots, \delta(q_{i_{n-1}}, a_n) = q_{i_n}$ com $q_n \in F, I$ sendo uma sequência 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 6.7 Considerando os Exemplos 6.5 e 6.6 tem-se que a palavra $aaaa$ não é aceita pelo AFD do Exemplo 6.5, uma vez que, $q_0 \notin F$. Já a palavra 0101 também não é aceita pelo AFD do Exemplo 6.6, uma vez que, $s_1 \notin F$, de fato o leitor atento pode notar que o AFD do Exemplo 6.6 não aceita qualquer palavra de entrada pois $F = \emptyset$.

Observação: Note que se um AFD não aceitar qualquer palavra, é diferente de dizer que ele não realiza computação, pois como já mencionado anteriormente uma computação é apenas a aplicação da função $\widehat{\delta}$, já não aceitar palavras diz respeito ao fato de $F = \emptyset$.

Exemplo 6.8 Considere o AFD representado pelo grafo de transições a seguir.

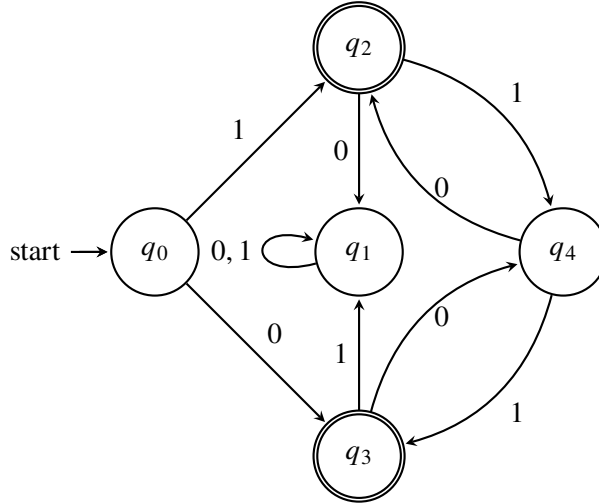


Figura 6.3: Um AFD com dois estados finais.

Por indução sobre o tamanho das palavras é fácil mostrar que este AFD reconhece palavras as palavras $1(10)^n1$ e $0(01)^n0$ 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 6.5 (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:

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \widehat{\delta}(q_0, w) \in F\} \quad (6.3)$$

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 6.9 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 6.4 a seguir.

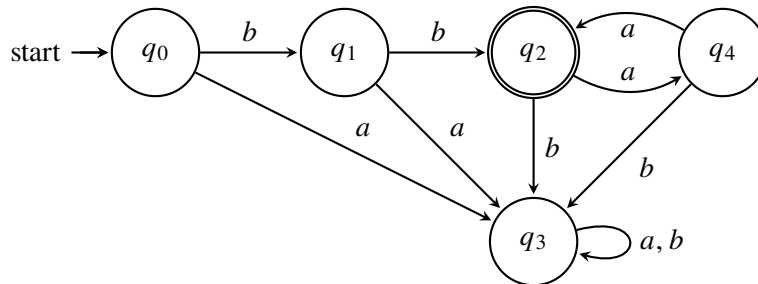


Figura 6.4: AFD A_1 que reconhece a linguagem $\{bba^{2n} \mid n \in \mathbb{N}\}$.

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

que,

- **Base da indução:**

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 $bb \in \mathcal{L}(A_1)$.

- **Hipótese indutiva (HI):**

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

- **Passo indutivo:**

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

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

Logo $\widehat{\delta}(q_0, bba^{2n}) \in \mathcal{L}(A_1)$.

(\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 6.4) 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 6.10 O AFD A do Exemplo 6.1 reconhece a linguagem $L = \{a^{2n+1} \mid n \in \mathbb{N}\}$.

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

- **Base da indução:**

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, $w \in \mathcal{L}(A)$.

- **Hipótese indutiva (HI):**

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

- **Passo indutivo:**

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

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

(\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 [34], 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 6.6 (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} . ♣

6.2 Autômato Finito Não-determinístico

Como explicado por Peter Linz em [35], um autômato finito não-determinístico, ou simplesmente AFN, é um autômato que se diferencia dos AFD apenas no quesito da função de transição. A diferença consiste no fato de que, enquanto a imagem da função de transição em um AFD é sempre um estado, nos AFN a imagem da função de transição é um subconjunto de estados, em um sentido moderno da teoria dos autômatos, um AFN seria uma máquina que algumas transições geraria uma superposição de estados [21]. Formalmente um AFN é como se segue.

Definição 6.7 (Autômato Finito Não-determinístico)

Um AFN é uma estrutura $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$ onde: Q, Σ, q_0 e F são da mesma forma que na Definição 6.1, já $\delta_N : Q \times \Sigma \rightarrow \wp(Q)$ é uma função total (chamada função de transição não determinística). ♣

Exemplo 6.11 A estrutura $A = \langle \{q_0, q_1, q_2\}, \{a, b\}, \delta_N, q_0, \{q_0, q_1\} \rangle$ onde a função δ é descrita pela Tabela 6.1 a seguir é um AFN.

$\begin{array}{c} \backslash \\ Q \end{array} \quad \Sigma$	a	b
q_0	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_0, q_2\}$
q_2	$\{q_2\}$	$\{q_1\}$

Tabela 6.1: Tabela de transição para a função δ_N do AFN no Exemplo 6.11.

A representação visual de um AFN usando grafos de transição é construída exatamente da mesma forma que a representação de um AFD, a diferença é o fato de poder existir múltiplas arestas rotuladas por um símbolo $a \in \Sigma$ saindo de um vértice q_i e chegando nos vértices $q_j \in X$ onde $X \subseteq Q$ e existe a transição $\delta_N(q_i, a) = X$.

Exemplo 6.12 O grafo de transição representado na Figura 6.5 a seguir é uma representação para o AFN do Exemplo 6.11.

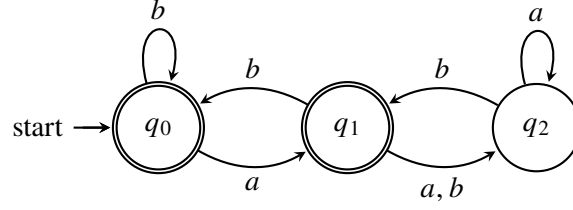


Figura 6.5: Grafo de transição do AFN do Exemplo 6.11.

Observação: Para as transições da forma $\delta_N(q_0, a) = \emptyset$ tem-se que as mesmas não são representadas no grafo de transição de um AFN.

Como para o caso determinístico a função de transição, neste caso δ_N , pode ser estendida para uma função $\widehat{\delta}_N$ usando recursividade como se segue.

Definição 6.8 (Transição não-determinística estendida)

Seja $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$ um AFN, a função de transição estendida é uma função $\delta_N : Q \times \Sigma^* \rightarrow \wp(Q)$ definida pela seguinte recursão.

$$\widehat{\delta}_N(q, \lambda) = \{q\} \quad (6.4)$$

$$\widehat{\delta}_N(q, wa) = \bigcup_{q' \in \widehat{\delta}_N(q, w)} \delta_N(q', a) \quad (6.5)$$

Como para os AFD a noção de computação em qualquer AFN consiste simplesmente da aplicação da função $\widehat{\delta}_N$ sobre alguma palavra $w \in \Sigma^*$ e um estado q .

Exemplo 6.13 Considerando o AFN ilustrado na Figura 6.5 e a palavra “abb” tem-se que,

$$\widehat{\delta}_N(q_0, abb) = \bigcup_{q' \in \widehat{\delta}_N(q_0, ab)} \delta_N(q', b) \quad (6.6)$$

mas tem-se que,

$$\widehat{\delta}_N(q_0, ab) = \bigcup_{q'' \in \widehat{\delta}_N(q_0, a)} \delta_N(q'', b) \quad (6.7)$$

e

$$\begin{aligned} \widehat{\delta}_N(q_0, a) &= \bigcup_{q''' \in \widehat{\delta}_N(q_0, \lambda)} \delta_N(q''', a) \\ &= \bigcup_{q''' \in \{q_0\}} \delta_N(q''', a) \\ &= \delta_N(q_0, a) \\ &= \{q_1\} \end{aligned} \quad (6.8)$$

substituindo a Equação (6.8) na Equação (6.7) tem-se que,

$$\widehat{\delta}_N(q_0, ab) = \{q_0, q_2\} \quad (6.9)$$

e finalmente substituindo a Equação (6.9) na Equação (6.6) tem-se que,

$$\widehat{\delta}_N(q_0, aba) = \{q_0, q_1\} \quad (6.10)$$

ou seja, a computação da palavra “abb” pelo AFN da Figura 6.5 termina no conjunto de estados $\{q_0, q_1\}$.

Pelo exemplo anterior o leitor mais atento pode ter notado que diferente do caso determinístico, a computação em um AFN não é linear, no sentido de que não existe um único caminho de computação¹, em vez disso, a computação em um AFN pode ser vista como uma árvore em que a união dos estados em cada nível da árvore representa a superposição de estados assumida pela unidade de controle do autômato a cada símbolo consumido (computado ou lido) da palavra w , o exemplo a seguir ilustra bem essa ideia de árvore de computação.

Exemplo 6.14 Considerando o AFN ilustrado na Figura 6.5 e a palavra “ $abab$ ” tem-se que o processo de computação para tal palavra poder ser representado pela árvore da Figura 6.6 a seguir.

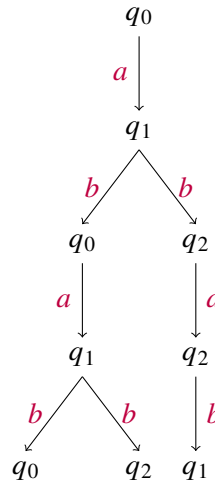


Figura 6.6: Árvore de computação da palavra “ $abab$ ” no AFN da Figura 6.5.

Pode-se agora apresentar a noção de aceitação (reconhecimento ou computação) de palavras nos AFD.

Definição 6.9 (Reconhecimento de palavras em AFN)

Sejam $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$ um AFN e seja $w \in \Sigma^*$. A palavra w é dita aceita (reconhece ou computada) por A sempre que $\widehat{\delta}(q_0, w) \cap F \neq \emptyset$ e é rejeitada por A em qualquer outro caso.

Note que a Definição 6.9 pode ser informalmente interpretada da seguinte forma, uma palavra é aceita por um AFN A se existe pelo menos um caminho de computação para w que termine em um estado final, isto é, pelo menos uma das folhas na árvore de computação deve ser um estado $q \in F$, neste caso w é aceita por A .

Exemplo 6.15 Considerando o AFN representado pela Figura 6.7 a seguir,

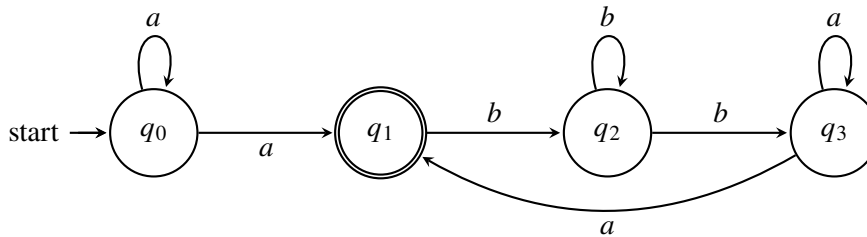


Figura 6.7: Grafo de transição de um AFN do Exemplo 6.15.

para as palavras “ $aabbbba$ ” e “ $aabb$ ” tem-se que:

$$\widehat{\delta}_N(q_0, aabbbba) = \{q_1, q_3\}$$

¹Como explicado em [21] um caminho de computação é uma sequência de estados assumidos pela unidade central do autômato durante o processamento de uma palavra de entrada.

e

$$\widehat{\delta}_N(q_0, aabb) = \{q_2, q_3\}$$

logo a palavra “aabbba” é aceita por tal AFN. Por outro lado, a palavra “aabb” não é aceita pelo AFN.

Usando a definição apresentada anteriormente de palavra aceita pode-se finalmente introduzir formalmente a noção de linguagem aceita (computada ou reconhecida) pelos AFN.

Definição 6.10 (Linguagem de um AFN)

Seja $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$ um AFN 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:

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \widehat{\delta}_N(q_0, w) \cap F \neq \emptyset\} \quad (6.11)$$

De forma similar ao que ocorre com os AFD, para mostrar que uma linguagem L é aceita por algum AFN A deve-se provar a igualdade $L = \mathcal{L}(A)$, ou seja, deve-se provar que $w \in L \iff w \in \mathcal{L}(A)$.

Exemplo 6.16 A linguagem $L = \{a^i(ba)^j \mid i \geq 1, j \geq 0\}$ é aceita pelo AFN A representado pelo grafo de transição da Figura 6.8 a seguir.

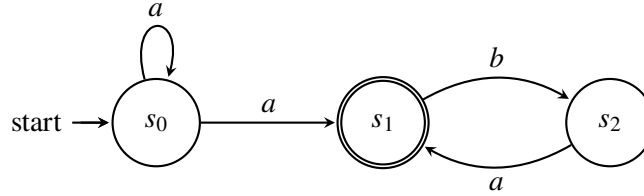


Figura 6.8: Grafo de transição de um AFN.

Demonstração (\Rightarrow) Suponha que $w \in L$, portanto, $w = a^m(ba)^n$, e agora por indução dupla sobre o par (m, n) tem-se que:

• **Base da indução:**

Quando com $m = 1$ e $n = 0$ vale a igualdade $w = a^1(ba)^0 = a$, agora usando a definição de δ_N do AFN A como representado na Figura 6.8 tem-se que,

$$\widehat{\delta}_N(s_0, a) = \bigcup_{s' \in \widehat{\delta}(s_0, a)} \delta_N(s', a) = \delta_N(s_0, a) = \{s_0, s_1\}$$

uma vez que, $s_1 \in F$ tem-se que $\widehat{\delta}_N(s_0, a) \cap F \neq \emptyset$, e portanto, $w \in \mathcal{L}(A)$. Agora suponha que para $w = a^1(ba)^n$ com $n \geq 0$ tem-se que $\widehat{\delta}_N(s_0, a^1(ba)^n) \cap F \neq \emptyset$. Assim dado $a^1(ba)^{n+1}$ por definição tem-se que:

$$\begin{aligned} \widehat{\delta}_N(s_0, a^1(ba)^{n+1}) &= \widehat{\delta}_N(s_0, a^1(ba)^n ba) \\ &= \bigcup_{s' \in \widehat{\delta}_N(s_0, a^1(ba)^n b)} \delta_N(s', a) \end{aligned} \quad (6.12)$$

agora fazendo,

$$K = \bigcup_{s'' \in \widehat{\delta}_N(s_0, a^1(ba)^n)} \delta_N(s'', b) \quad (6.13)$$

e reescrevendo a Equação (6.12) usando a Equação (6.13) tem-se que,

$$\widehat{\delta}_N(s_0, a^1(ba)^{n+1}) = \bigcup_{s' \in K} \delta_N(s', a) \quad (6.14)$$

entretanto, por hipótese tem-se que $\widehat{\delta}_N(s_0, a^1(ba)^n) \cap F \neq \emptyset$, consequentemente, tem-se que $s_1 \in$

$\widehat{\delta}_N(s_0, a^1(ba)^n)$ dessa forma pela Equação (6.13) é claro que $\delta_N(s_1, b) \subseteq K$. Mas $\delta_N(s_1, b) = \{s_2\}$ logo pela Equação (6.14) tem-se que $\delta_N(s_2, a) \subseteq \widehat{\delta}_N(s_0, a^1(ba)^{n+1})$, desde que $\delta_N(s_2, a) = \{s_1\}$, tem-se $s_1 \in \widehat{\delta}_N(s_0, a^1(ba)^{n+1})$, portanto, $\widehat{\delta}_N(s_0, a^1(ba)^{n+1}) \cap F \neq \emptyset$, consequentemente $a^1(ba)^{n+1} \in \mathcal{L}(A)$.

• **Hipótese indutiva (HI):**

Assuma que para todo $n \geq 0$ tem-se que $\widehat{\delta}_N(s_0, a^m(ba)^n) \cap F \neq \emptyset$.

• **Passo indutivo:**

Primeiro seja $w \in L$ de forma que $w = a^{m+1}(ba)^0$ logo pela hipótese indutiva segue que,

$$\widehat{\delta}_N(s_0, a^{m+1}(ba)^0) \cap F \neq \emptyset$$

consequentemente, $a^{m+1}(ba)^0 \in \mathcal{L}(A)$. Por outro lado, sendo $w \in L$ tal que $w = a^{m+1}(ba)^n$, usando a definição de $\widehat{\delta}_N$ tem-se para $a^{m+1}(ba)^{n+1}$ que,

$$\begin{aligned} \widehat{\delta}_N(s_0, a^{m+1}(ba)^{n+1}) &= \widehat{\delta}_N(s_0, a^{m+1}(ba)^n ba) \\ &= \bigcup_{s' \in \widehat{\delta}_N(s_0, a^{m+1}(ba)^n b)} \delta_N(s', a) \end{aligned} \quad (6.15)$$

agora desenvolvendo o termo $\widehat{\delta}_N(s_0, a^{m+1}(ba)^n b)$ tem-se

$$\widehat{\delta}_N(s_0, a^{m+1}(ba)^n b) = \bigcup_{s'' \in \widehat{\delta}_N(s_0, a^{m+1}(ba)^n)} \delta_N(s'', b)$$

pela hipótese indutiva tem-se que $\widehat{\delta}_N(s_0, a^{m+1}(ba)^n) \cap F = \emptyset$, consequentemente, $s_1 \in \widehat{\delta}_N(s_0, a^{m+1}(ba)^n)$, logo $\delta_N(s_1, b) \subseteq \widehat{\delta}_N(s_0, a^{m+1}(ba)^n)$, uma vez que, $\delta_N(s_1, b) = \{s_2\}$, tem-se que $\{s_2\} \subseteq \widehat{\delta}_N(s_0, a^{m+1}(ba)^n)$, assim pela Equação (6.15) segue que $\delta_N(s_2, a) \subseteq \widehat{\delta}_N(s_0, a^{m+1}(ba)^{n+1})$, mas por definição $\delta_N(s_2, a) = \{s_1\}$, portanto, tem-se que $\{s_1\} \subseteq \widehat{\delta}_N(s_0, a^{m+1}(ba)^{n+1})$, logo $\widehat{\delta}_N(s_0, a^{m+1}(ba)^{n+1}) \cap F \neq \emptyset$ e assim $a^{m+1}(ba)^{n+1} \in \mathcal{L}(A)$.

(\Leftarrow) Suponha que $w \in \mathcal{L}(A)$ assim $s_1 \in \widehat{\delta}_N(s_0, w)$, note porém que s_1 só é acessível a partir de duas transições:

- (1) $\delta_N(s_0, a)$ e
- (2) $\delta_N(s_2, a)$.

Note que devido ao *loop* fornecido pelo fato de que $s_0 \in \delta_N(s_0, a)$ a transição (1) pode ser executada m vezes com $m \geq 1$, em que para cada execução um novo ramo com o estado s_1 é gerado na árvore de computação de A , entretanto, executar m vezes a transição $\delta_N(s_0, a)$ implica em executar a computação $\widehat{\delta}_N(s_0, a^m)$, pelo fato² de que $s_1 \in \widehat{\delta}_N(s_0, a^m)$ tem-se que $a^m \in \mathcal{L}(A)$, e uma vez que $a^m = a^m(ba)^0$ tem-se que a primeira forma de $\widehat{\delta}_N(s_0, w) \cap F \neq \emptyset$ é que $w = a^m(ba)^0$ e assim $w \in L$. Por outro lado, para acessar s_1 via a transição (2) é necessário antes chegar a um ramo de computação em que o estado s_2 seja uma folha, mas pela definição de A isso só é possível se a transição $\delta_N(s_1, b)$ for usada, note entretanto, que as transições $\delta_N(s_1, b) = \{s_2\}$ e $\widehat{\delta}_N(s_2, a) = \{s_1\}$ também geram um *loop* que pode ser executado n vezes com $n \geq 0$, mas executar esse *loop* n vezes corresponde a executar $\widehat{\delta}_N(s_1, (ba)^n)$, e como dito anteriormente, s_1 só é acessível pela definição de A usando a computação $\widehat{\delta}_N(s_0, a^m)$, portanto, para que $s_1 \in \widehat{\delta}_N(s_0, w) \cap F$, obrigatoriamente, $w = a^m(ba)^n$ com $m \geq 1, n \geq 0$, e portanto, $w \in L$. ■

Exemplo 6.17 O AFN S representado no grafo de transição exposto na Figura 6.9 a seguir reconhece a linguagem $L = \{uv \mid u \in \{0, 1\}^*, v \in \{0, 1\}\}$.

Demonstração (\Rightarrow) A ida fica a cargo do leitor. (\Leftarrow) Suponha que $w \in \mathcal{L}(A)$ assim por definição $\widehat{\delta}_N(s_0, w) \cap \{s_1, s_2\} \neq \emptyset$, agora pela definição de δ_N é claro que toda árvore de computação de A apresenta a propriedade de

²Fica para o leitor a tarefa de provar que para todo $m \geq 1$ tem-se que $s_1 \in \widehat{\delta}_N(s_0, a^m)$.

sempre conter um dos estados s_1 ou s_2 , mas nunca os dois simultaneamente³, além disso, o fato de $s_0 \in \widehat{\delta_N}(s_0, a)$ para todo $a \in \{0, 1\}$, garante que qualquer palavra não vazia u sobre o alfabeto $\{0, 1\}$ pode ser gerada, por fim, no último passo de computação é claro que s_1 ou s_2 será uma folha da árvore, entretanto, s_1 só será tal folha no caso da palavra terminar em 0 caso contrário a folha será s_2 , e portanto, todo $w \in \mathcal{L}(A)$ tem a forma uv com $u \in \{0, 1\}^*$ e $v \in \{0, 1\}$, consequentemente $w \in L$. ■

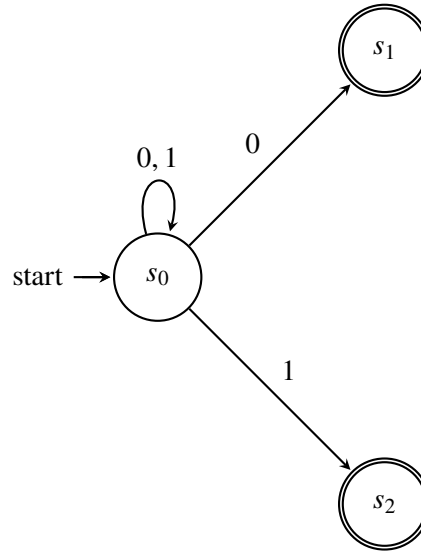


Figura 6.9: Grafo de transição de um AFN S do Exemplo 6.17.

De forma ingênua o leitor pode vir a imaginar que a possibilidade da unidade de controle de um AFN poder assumir mais de um estado interno simultaneamente, faz com que os AFN sejam mais poderosos que os AFD, entretanto, como será exibido pelos resultados a seguir, isso não ocorre, de fato, como dito [12, 35] apesar de tornar mais fácil a tarefa de construir um autômato quem reconheça uma linguagem L , o não-determinismo não aumenta nem nada o poder de computação dos autômatos finitos.

Teorema 6.1 (Transformação AFD - AFN)

Se $L = \mathcal{L}(A)$ para algum AFD A , então existe um AFN A' tal que $L = \mathcal{L}(A')$.

Demonstração A prova é trivial, uma vez que, todo AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ pode ser convertido em um AFN $A' = \langle Q, \Sigma, \delta_N, q_0, F \rangle$ apenas realizando as transformações das transições $\delta(q_i, a) = q_j$ nas transições não-determinísticas $\delta_N(q_i, a) = \{q_j\}$ e mantendo todo o resto da estrutura igual. ■

O próximo resultado estabelece a contraparte do Teorema 6.1, isto é, tal resultado mostrará que sempre é possível obter um AFD que pode “simular”⁴ um AFN.

Teorema 6.2 (Transformação AFN - AFD)

Se $L = \mathcal{L}(A)$ para algum AFN A , então existe um AFD A' tal que $L = \mathcal{L}(A')$.

Demonstração Suponha que $L = \mathcal{L}(A)$ para algum AFN $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$, agora é construído um autômato $A' = \langle \wp(Q), \Sigma, \delta, \{q_0\}, F' \rangle$ onde para todo $X \in \wp(Q)$ e $a \in \Sigma$ tem-se

$$\delta(X, a) = \bigcup_{q \in X} \delta_N(q, a) \quad (6.16)$$

³A prova desta propriedade fica como exercício ao leitor.

⁴O termo simular aqui, diz respeito a ideia de que cada aplicação de um função de transição não-determinística pode ser representada de forma precisa por uma aplicação de uma função de transição determinística, para detalhes consulte [33, 45].

claramente este autômato é realmente determinístico, e para todo $X \in \wp(Q)$ tem-se que $X \in F'$ se, e somente se, $X \cap F \neq \emptyset$. Agora será mostrado por indução sobre o tamanho de $w \in \Sigma^*$ que:

$$\widehat{\delta}(\{q_0\}, w) = \widehat{\delta}_N(q_0, w)$$

- **Base da indução:**

Quando $|w| = 0$ isto é $w = \lambda$ tem-se trivialmente pela definição das funções de transição estendidas que $\widehat{\delta}(\{q_0\}, \lambda) = \widehat{\delta}_N(q_0, \lambda)$.

- **Hipótese indutiva (HI):**

Suponha que para todo $w \in \Sigma^*$ com $|w| \geq 0$ tem-se que $\widehat{\delta}(\{q_0\}, w) = \widehat{\delta}_N(q_0, w)$.

- **Passo indutivo:**

Dado $w = ua$ com $u \in \Sigma^*$, $|u| \geq 0$ e $a \in \Sigma$ tem-se que,

$$\begin{aligned} \widehat{\delta}(\{q_0\}, w) &= \widehat{\delta}(\{q_0\}, ua) \\ &= \delta(\widehat{\delta}(\{q_0\}, u), a) \\ &\stackrel{(HI)}{=} \delta(\widehat{\delta}_N(q_0, u), a) \\ &\stackrel{Eq.(6.16)}{=} \bigcup_{q \in \widehat{\delta}_N(q_0, u)} \delta_N(q, a) \\ &= \widehat{\delta}_N(q_0, ua) \\ &= \widehat{\delta}_N(q_0, w) \end{aligned}$$

Portanto, pode-se concluir que $w \in \mathcal{L}(A)$ se, e somente se, $w \in \mathcal{L}(A')$, ou seja, $L = \mathcal{L}(A')$ o que completa a prova. ■

Observe que o método de construção usado na prova do Teorema 6.2 cria um AFD cujo número de estado cresce em razão de uma potência de 2 quando comparado com o quantitativo de estados do AFN original. Como consequência deste resultado segue o seguinte corolário.

Corolário 6.1

Uma linguagem L é regular se, e somente se, existe um AFN A tal que $L = \mathcal{L}(A)$.

Demonstração (\Rightarrow) Assuma que L é regular, assim por definição existe um AFD A' tal que $L = \mathcal{L}(A')$, entretanto, pelo Teorema 6.1 existe um AFN A tal que $L = \mathcal{L}(A)$. (\Leftarrow) Suponha que $L = \mathcal{L}(A)$ para algum AFN A , agora pelo Teorema 6.2 existe um AFD A' tal que $L = \mathcal{L}(A')$, e portanto, por definição L é regular. ■

É importante destacar que o método de construção do AFD usado na prova do Teorema 6.2, conhecido como método de construção das partes introduzido por Rabin e Scott em [52], tem a característica de poder vim a produzir durante sua execução alguns estados inacessíveis⁵ no AFD resultante.

Outro ponto sobre o método de construção das partes é que em alguns cenários pode ser tornar impraticável, pois se o AFN de entrada possuir n estados, o AFD resultante do método terá 2^n estados, ou seja, o crescimento no número de estados do AFD resultante do método cresce proposicional a uma potência de 2, o que rapidamente gera um número exponencialmente grande de estados.

A seguir o leitor será apresentado a uma melhoria no algoritmo de construção das partes, no sentido de que, a execução de tal algoritmo não produz estados inacessíveis no AFD de saída, o algoritmo a seguir é um pseudo-código baseado na versão textual apresentada no livro de Bedregal *et al.* [12]. A melhoria no Algoritmo 1 consiste do fato dele não considerar simplesmente o conjunto $\wp(Q)$ no AFD de saída, em vez disso, ele

⁵Um estado q em um AFD é dito inacessível se não existe um $w \in \Sigma^*$ tal que $\widehat{\delta}(q_0, w) = q$. Vale também ressaltar como destaque em [12, 33] que estados inacessíveis não aumentam o poder de computação nos AFD.

constrói iterativamente um conjunto de estados $Q' \subseteq \wp(Q)$, que no pior caso⁶ $Q' = \wp(Q)$.

Algoritmo 1: Algoritmo para converter AFN em AFD sem estados inacessíveis.

Entrada: Um AFN $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$
Saída: Um AFD $A' = \langle Q', \Sigma, \delta, \{q_0\}, F' \rangle$

```

1  início
2  | Inicialize o conjuntos  $Q_u$  com um estado rotulado por  $\{q_0\}$ 
3  | Inicialize o conjuntos  $Q'$  com um estado rotulado por  $\{q_0\}$ 
4  | Inicialize o conjunto  $F'$  como sendo vazio
5  | repita
6  | | Selecione um estado  $X \in Q_u$ 
7  | | para cada  $a \in \Sigma$  faça
8  | | | Determine o conjunto  $Y = \bigcup_{q \in X} \delta_N(q, a)$ 
9  | | | se  $Y \notin Q'$  então
10 | | | | Adicione um estado rotulado por  $Y$  em  $Q'$ 
11 | | | | Adicione um estado rotulado por  $Y$  em  $Q_u$ 
12 | | | | Defina a transição  $\delta(X, a) = Y$ 
13 | | | senão
14 | | | | Defina a transição  $\delta(X, a) = Y$ 
15 | | | fim
16 | | fim
17 | | Remova  $X$  de  $Q_u$ 
18 | até  $Q_u = \emptyset$ ;
19 | para cada  $X \in Q'$  faça
20 | | se  $X \cap F \neq \emptyset$  então
21 | | | Adicione  $X$  ao conjunto  $F'$ 
22 | | fim
23 | fim
24 | retorna  $A' = \langle Q', \Sigma, \delta, \{q_0\}, F' \rangle$ 
25 fim
```

Exemplo 6.18 Usando o Algoritmo 1 tendo o AFN representado pelo grafo de transição da Figura 6.7 como entrada será obtido o AFD $M = \langle \{s_0, s_1, s_2, s_3, s_4, \emptyset\}, \{a, b\}, \delta, s_0, F' \rangle$ onde tem-se os estados equivalem aos seguintes conjuntos $s_0 = \{q_0\}$, $s_1 = \{q_0, q_1\}$, $s_2 = \{q_2\}$, $s_3 = \{q_2, q_3\}$, $s_4 = \{q_1, q_3\}$ tem-se que $F' = \{s_1, s_4\}$ e a função de transição δ é como se segue.

$$\delta(s_0, a) = s_1, \delta(s_1, a) = s_1, \delta(s_2, a) = \emptyset, \delta(s_3, a) = s_4, \delta(s_4, a) = s_4, \delta(\emptyset, a) = \emptyset$$

$$\delta(s_0, b) = \emptyset, \delta(s_1, b) = s_2, \delta(s_2, b) = s_3, \delta(s_3, b) = s_3, \delta(s_4, b) = s_2, \delta(\emptyset, b) = \emptyset$$

6.3 Autômatos Finitos Não-determinísticos com Movimentos Vazios

Os λ -Autômatos Finitos Não-determinísticos, ou simplesmente, λ -AFN são como dito em [45], uma generalização do modelo de AFN que foi introduzido na seção anterior e que são permitidas transições entre estados diferentes usando (ou consumindo) a palavra vazia, tais transições recebem o nome de λ -transições, a seguir tais autômatos serão apresentados formalmente.

⁶A expressão “no pior caso” é típica da análise de algoritmos, em momentos futuros essa ideia de pior caso será melhor desenvolvida neste manuscrito.

Definição 6.11 (λ -Autômatos Finitos Não-determinísticos)

Um λ -AFN é uma estrutura $A = \langle Q, \Sigma, \underline{\delta}_N, q_0, F \rangle$ onde: Q, Σ, q_0 e F são da mesma forma que na Definição 6.1, já $\underline{\delta}_N : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \wp(Q)$ é uma função total (chamada λ -função de transição não determinística).

A representação usando grafos de transição dos λ -AFN é similar a representação dos AFN da seção anterior, a única diferença é que podem haver transições rotuladas pelo símbolo λ , isto é, podem existir no grafo arestas entre vértices que são rotuladas por λ , e o mesmo vale para a representação das árvores de computação.

Exemplo 6.19 A estrutura $A = \langle \{q_0, q_1, q_2\}, \{0, 1\}, \underline{\delta}_N, q_0, \{q_0\} \rangle$ com $\underline{\delta}_N$ sendo especificada pela Tabela 6.2 a seguir é um λ -AFN.

$\Sigma \cup \{\lambda\}$	0	1	λ
Q			
q_0	\emptyset	\emptyset	$\{q_1\}$
q_1	$\{q_1\}$	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$	$\{q_0, q_2\}$

Tabela 6.2: Tabela de transição para a função δ_N do AFN no Exemplo 6.19.

Observação: Uma interpretação para as transições da forma $\underline{\delta}_N(q, \lambda) = X$ é que a unidade de controle do autômato consegue mudar seu estado interno q para um subconjunto de estados X sem precisar acessar a memória.

Exemplo 6.20 O grafo de transição representado na Figura 6.10 a seguir é uma representação para o λ -AFN do Exemplo 6.19.

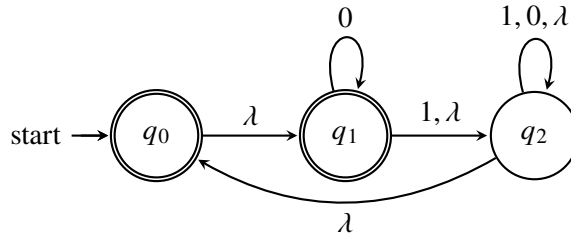


Figura 6.10: Grafo de transição do λ -AFN do Exemplo 6.19.

Note porém que a definição da função de transição $\underline{\delta}_N$ garante que as transições em um λ -AFN acontecem apenas em duas situações, a primeira em relação símbolos individuais do alfabeto Σ e a segunda com relação a palavra vazia, assim não existe uma forma de computar uma palavra w de forma que $|w| > 1$. A saída para contorna esse fato é estender a função de transição do autômato, similarmente ao que é feito para os AFD e AFN, para isso entretanto, é necessária algumas definições adicionais.

Definição 6.12 (Função δ_λ)

Seja $A = \langle Q, \Sigma, \underline{\delta}_N, q_0, F \rangle$ um λ -AFN, então a função $\delta_\lambda : Q \rightarrow \wp(Q)$ é definida como,

$$\delta_\lambda(q) = \bigcup_{i=0}^n \lambda\text{-fecho}^i(q) \quad (6.17)$$

onde $n = \#Q - 1$ e

$$\lambda\text{-fecho}^0(q) = \{q\} \quad (6.18)$$

$$\lambda\text{-fecho}^i(q) = \bigcup_{q' \in \lambda\text{-fecho}^{i-1}(q)} \delta_N(q', \lambda) \quad (6.19)$$

Exemplo 6.21 Considere o λ -AFN da Figura 6.10 tem-se para o estado q_1 que,

$$\lambda\text{-fecho}^2(q_1) = \bigcup_{q' \in \lambda\text{-fecho}^1(q_1)} \delta_N(q', \lambda) \quad (6.20)$$

desenvolvendo $q' \in \lambda\text{-fecho}^1(q_0)$ tem-se que,

$$\lambda\text{-fecho}^1(q_1) = \bigcup_{q' \in \lambda\text{-fecho}^0(q_1)} \delta_N(q', \lambda)$$

mas,

$$\lambda\text{-fecho}^0(q_1) = \{q_1\}$$

assim,

$$\lambda\text{-fecho}^1(q_1) = \{q_2\}$$

substituindo tal resultado na Equação 6.20 tem-se que,

$$\begin{aligned} \lambda\text{-fecho}^2(q_1) &= \bigcup_{q' \in \{q_2\}} \delta_N(q', \lambda) \\ &= \{q_2, q_0\} \end{aligned}$$

logo $\delta_\lambda(q_1) = \{q_0, q_1, q_2\}$.

Uma interpretação semântica para a função δ_λ é que ela representa a resposta ao questionamento: “Estando no estado q e executando n λ -transições qual subconjunto de estados a unidade central do autômato irá assumir?”. Assim como acontecer com as funções de transição a função δ_λ pode ser estendida, a seguir é exposto tal extensão.

Definição 6.13 (Função $\widehat{\delta}_\lambda$)

Seja $A = \langle Q, \Sigma, \underline{\delta}_N, q_0, F \rangle$ um λ -AFN, então a função $\widehat{\delta}_\lambda : \wp(Q) \rightarrow \wp(Q)$ é definida como,

$$\widehat{\delta}_\lambda(X) = \bigcup_{q \in X} \delta_\lambda(q) \quad (6.21)$$

Exemplo 6.22 Considere o λ -AFN da Figura 6.10 tem-se para o conjunto $\{q_1, q_2\}$ que,

$$\begin{aligned} \widehat{\delta}_\lambda(\{q_1, q_2\}) &= \bigcup_{q \in \{q_1, q_2\}} \delta_\lambda(q) \\ &= \delta_\lambda(q_1) \cup \delta_\lambda(q_2) \\ &= \{q_0, q_1, q_2\} \cup \{q_0, q_2, q_1\} \\ &= \{q_0, q_2, q_1\} \end{aligned}$$

Agora usando as definições de δ_λ e $\widehat{\delta}_\lambda$ pode-se apresentar a extensão da função de transição dos λ -AFN.

Definição 6.14 (λ -Transição não-determinística estendida)

Seja $A = \langle Q, \Sigma, \underline{\delta}_N, q_0, F \rangle$ um λ -AFN a função $\underline{\delta}_N$ é estendido para a função $\widehat{\underline{\delta}}_N : Q \times \Sigma^* \rightarrow \wp(Q)$ definida pela seguinte recursão:

$$\widehat{\underline{\delta}}_N(q, \lambda) = \delta_\lambda(q) \quad (6.22)$$

$$\widehat{\underline{\delta}}_N(q, wa) = \bigcup_{q' \in \widehat{\underline{\delta}}_N(q, w)} \widehat{\delta}_\lambda(\underline{\delta}_N(q', a)) \quad (6.23)$$

Exemplo 6.23 Considere o λ -AFN da Figura 6.10 tem-se a seguinte computação para a palavra “10”:

$$\begin{aligned}\widehat{\delta_N}(q_0, 10) &= \bigcup_{q' \in \widehat{\delta_N}(q_0, 1)} \widehat{\delta_\lambda}(\delta_N(q', 0)) \\ &= \bigcup_{q' \in \widehat{\delta_N}(q_0, 1)} \widehat{\delta_\lambda}(\delta_N(q', 0))\end{aligned}\quad (6.24)$$

mas,

$$\begin{aligned}\widehat{\delta_N}(q_0, 1) &= \bigcup_{q' \in \widehat{\delta_N}(q_0, \lambda)} \widehat{\delta_\lambda}(\delta_N(q', 1)) \\ &= \bigcup_{q' \in \delta_\lambda(q_0)} \widehat{\delta_\lambda}(\delta_N(q', 1)) \\ &= \bigcup_{q' \in \{q_0, q_1, q_2\}} \widehat{\delta_\lambda}(\delta_N(q', 1)) \\ &= \widehat{\delta_\lambda}(\{q_1, q_2\}) \\ &= \{q_0, q_1, q_2\}\end{aligned}\quad (6.25)$$

substituindo o valor da Equação (6.25) na Equação (6.24) tem-se que,

$$\begin{aligned}\widehat{\delta_N}(q_0, 10) &= \bigcup_{q' \in \{q_0, q_1, q_2\}} \widehat{\delta_\lambda}(\delta_N(q', 0)) \\ &= \{q_0, q_1, q_2\}\end{aligned}$$

Assim como para o caso dos AFN uma palavra qualquer $w \in \Sigma^*$ será dita aceita por um λ -AFN quando a computação da palavra w para em pelo menos um estado final, ou seja, w é reconhecida pelo λ -AFN sempre que $\widehat{\delta_N}(q_0, w) \cap F \neq \emptyset$, e assim pode-se definir formalmente a noção de linguagem para os λ -AFN.

Definição 6.15 (Linguagem de um λ -AFN)

Seja $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$ um λ -AFN a linguagem aceita por A , denotado por $\mathcal{L}(A)$, corresponde ao seguinte conjunto.

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \widehat{\delta_N}(q_0, w) \cap F \neq \emptyset\} \quad (6.26)$$

Os aspectos relacionados a mostrar que um λ -AFN reconhece uma linguagem L são similares ao mesmo aspectos com respeito aos AFN.

Exemplo 6.24 O λ -AFN representado pelo grafo de transição da Figura 6.11 a seguir reconhece a linguagem $L = \{w \in \{1, 2, 3\}^* \mid w = 1^i 2^j 3^k \text{ com } i, j, k \in \mathbb{N}\}$.

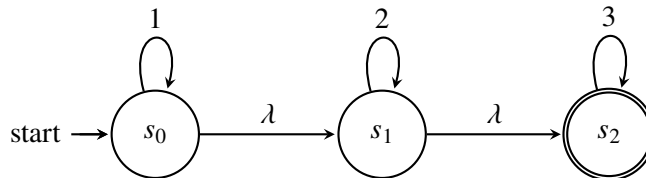
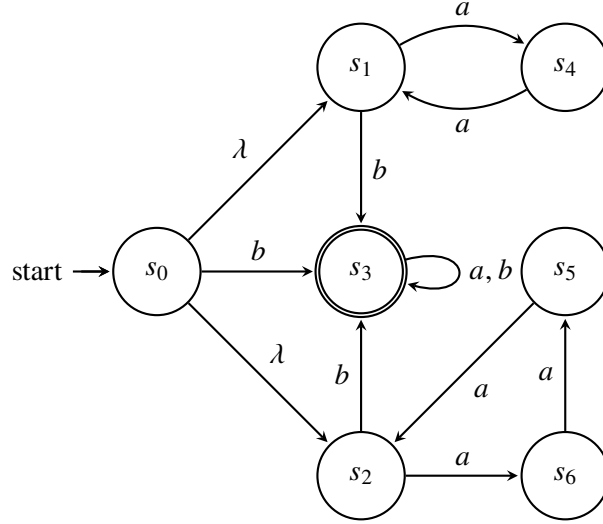


Figura 6.11: Grafo de transição do λ -AFN do Exemplo 6.24.

Exemplo 6.25 O λ -AFN representado pelo grafo de transição esboçado pela Figura 6.12, aceita a linguagem $L = \{uv \mid u \in \{a\}^*, |u|_a = 2k \text{ ou } |u|_a = 3k, v = bx, x \in \{a, b\}^*, k \in \mathbb{N}\}$.


 Figura 6.12: Grafo de transição do λ -AFN do Exemplo 6.25.

Teorema 6.3 (Transformação λ -AFN-AFD)

Se $L = \mathcal{L}(A)$ para algum λ -AFN A , então existe um AFD A' tal que $L = \mathcal{L}(A')$.

Demonstração Suponha que $L = \mathcal{L}(A)$ para algum λ -AFN $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$ agora defina o seguinte o autômato $A' = \langle \wp(Q), \Sigma, \delta, \delta_\lambda(q_0), F' \rangle$ onde para todo $X \in \wp(Q)$ tem-se que $X \in F'$ se, e somente se, $X \cap F \neq \emptyset$, e além disso, para todo $X \in \wp(Q)$ e $a \in \Sigma$ tem-se:

$$\delta(X, a) = \bigcup_{q \in X} \widehat{\delta_\lambda}(\delta_N(q, a)) \quad (6.27)$$

por essa construção obviamente esse autômato é um AFD⁷. Agora será mostrado por indução sobre o tamanho de $w \in \Sigma^*$ que:

$$\widehat{\delta}(\delta_\lambda(q_0), w) = \widehat{\delta_N}(q_0, w)$$

- **Base da indução:**

Quando $|w| = 0$ isto é $w = \lambda$ tem-se trivialmente pela definição das funções de transição estendidas que,

$$\begin{aligned} \widehat{\delta}(\delta_\lambda(q_0), \lambda) &= \delta_\lambda(q_0) \\ &= \widehat{\delta_N}(q_0, \lambda) \end{aligned}$$

- **Hipótese indutiva (HI):**

Suponha que para todo $w \in \Sigma^*$ com $|w| \geq 0$ tem-se que $\widehat{\delta}(\delta_\lambda(q_0), w) = \widehat{\delta_N}(q_0, w)$.

- **Passo indutivo:**

⁷A prova desse fato fica como exercício ao leitor.

Dado $w = ua$ com $u \in \Sigma^*$, $|u| \geq 0$ e $a \in \Sigma$ tem-se que,

$$\begin{aligned}
 \widehat{\delta}(\delta_\lambda(q_0), w) &= \widehat{\delta}(\delta_\lambda(q_0), ua) \\
 &= \delta(\widehat{\delta}(\delta_\lambda(q_0), u), a) \\
 &\stackrel{(HI)}{=} \delta(\widehat{\delta_N}(q_0, w), a) \\
 &\stackrel{Eq.(6.27)}{=} \bigcup_{q \in \widehat{\delta_N}(q_0, w)} \widehat{\delta}_\lambda(\delta_N(q, a)) \\
 &= \widehat{\delta_N}(q_0, ua) \\
 &= \widehat{\delta_N}(q_0, w)
 \end{aligned}$$

Portanto, pode-se concluir que $w \in \mathcal{L}(A)$ se, e somente se, $w \in \mathcal{L}(A')$, ou seja, $L = \mathcal{L}(A')$ o que completa a prova. ■

Teorema 6.4 (Transformação AFD- λ -AFN)

Se $L = \mathcal{L}(A)$ para algum AFD A , então existe um λ -AFN A' tal que $L = \mathcal{L}(A')$. ♡

Demonstração Trivial e ficará como exercício ao leitor. ■

Corolário 6.2

Uma linguagem L é regular se, e somente se, existe um λ -AFN A tal que $L = \mathcal{L}(A)$. ♡

Demonstração (\Rightarrow) Assuma que L é regular, assim por definição existe um AFD A' tal que $L = \mathcal{L}(A')$, entretanto, pelo Teorema 6.4 existe um λ -AFN A tal que $L = \mathcal{L}(A)$. (\Leftarrow) Suponha que $L = \mathcal{L}(A)$ para algum λ -AFN A , agora pelo Teorema 6.3 existe um AFD A' tal que $L = \mathcal{L}(A')$, e portanto, L é regular. ■

Observação: Note que o Corolário 6.2 estabelece que a existência de λ -transições não aumenta o poder de computação dos autômatos finitos.

Assim como para o caso da transformação de AFN em AFD, o processo de usar a construção do conjunto das partes no Teorema 6.3 possui a desvantagem de gera estados inacessíveis. Mas como discutido em [10, 11, 33, 35], algumas simples modificações no Algoritmo 1 fazem com que o novo algoritmo gerado seja capaz de remover as λ -transições e não sejam produzidos estados inacessíveis a seguir é apresentado este novo algoritmo.

Exemplo 6.26 Aplicando o Algoritmo 2 ao λ -AFN do Exemplo 6.25 é obtido como saída o AFD $D = \langle \{A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, \emptyset\}, \{a, b\}, \delta, \{s_0, s_1, s_2\}, F' \rangle$ onde tem-se $A_0 = \{s_0, s_1, s_2\}$, $A_1 = \{s_4, s_6\}$, $A_2 = \{s_3\}$, $A_3 = \{s_1, s_5\}$, $A_4 = \{s_2, s_4\}$, $A_5 = \{s_1, s_6\}$, $A_6 = \{s_4, s_5\}$ e $A_7 = \{s_1, s_2\}$ com $F' = \{A_2\}$ e δ é descrito a seguir.

$$\begin{aligned}
 \delta(A_0, a) &= A_1, \delta(A_0, b) = A_2, \delta(A_1, a) = A_3, \\
 \delta(A_1, b) &= \emptyset, \delta(A_2, a) = A_2, \delta(A_2, b) = A_2, \\
 \delta(A_3, a) &= A_4, \delta(A_3, b) = A_2, \delta(A_4, a) = A_5, \\
 \delta(A_4, b) &= A_2, \delta(A_5, a) = A_6, \delta(A_5, b) = A_2, \\
 \delta(A_6, a) &= A_7, \delta(A_6, b) = \emptyset, \delta(A_7, a) = A_1, \\
 \delta(A_7, b) &= A_2, \delta(\emptyset, a) = A_7, \delta(\emptyset, b) = \emptyset
 \end{aligned}$$

Observação: Argumentações sobre a corretude e a completude do Algoritmo 2 podem ser consultadas em [33].

Algoritmo 2: Algoritmo para remoção de λ -transições de um λ -AFN.

Entrada: Um λ -AFN $A = \langle Q, \Sigma, \delta_N, q_0, F \rangle$
Saída: Um AFD $A' = \langle Q', \Sigma, \delta, \delta_\lambda(q_0), F' \rangle$

```

1  início
2  | Inicialize o conjuntos  $Q_u$  com um estado rotulado por  $\delta_\lambda(q_0)$ 
3  | Inicialize o conjuntos  $Q'$  com um estado rotulado por  $\delta_\lambda(q_0)$ 
4  | Inicialize o conjunto  $F'$  como sendo vazio
5  repita
6  |   Selecione um estado  $X \in Q_u$ 
7  |   para cada  $a \in \Sigma$  faça
8  |       | Determine o conjunto  $Y = \widehat{\delta}_\lambda\left(\bigcup_{q \in X} \delta_N(q, a)\right)$ 
9  |       | se  $Y \notin Q'$  então
10 |       |     Adicione um estado rotulado por  $Y$  em  $Q'$ 
11 |       |     Adicione um estado rotulado por  $Y$  em  $Q_u$ 
12 |       |     Defina a transição  $\delta(X, a) = Y$ 
13 |       | senão
14 |       |     Defina a transição  $\delta(X, a) = Y$ 
15 |       fim
16 |   fim
17 |   Remova  $X$  de  $Q_u$ 
18 até  $Q_u = \emptyset$ ;
19 para cada  $X \in Q'$  faça
20 |   se  $X \cap F \neq \emptyset$  então
21 |       | Adicione  $X$  ao conjunto  $F'$ 
22 |   fim
23 fim
24 retorna  $A' = \langle Q', \Sigma, \delta, \delta_\lambda(q_0), F' \rangle$ 
25 fim
```



Nota: Nestas últimas seções foram usados os símbolos δ , δ_N e δ_N para denotar as funções de transições dos AFD, AFN e λ -AFN respectivamente, entretanto, isso foi feito apenas para tornar o texto mais didático e ajudar na conversão entre os tipos de autômatos, mas é comum encontrar na literatura (ver [12, 33, 35]) que independente do tipo de autômato sua função de transição é denotada apenas por δ .

6.4 Teorema Myhill-Nerode e a Minimização de AFD

Até agora este manuscrito se preocupou com a tarefa de saber se uma linguagem pode ou não ser reconhecida por um autômato finito, seja ele determinístico ou não-determinístico. Nesta seção será apresentada ao leitor a questão de eficiência no reconhecimento de linguagens em relação aos autômatos finitos, aqui será mostrado que o problema de encontrar um menor AFD que reconhece uma linguagem L é decidível.

Na teoria dos autômatos quando se usa a palavra “menor”, se está querendo dizer simplesmente aquele com o menor número possível de estados, ou seja, o AFD mínimo. Mais adiante será aqui provado, que esse AFD mínimo é único a menos de isomorfismo, ou seja, se dois AFD reconhecem a mesma linguagem, cada um tendo o menor número possível de estados, então eles são isomórficos. Isso significa que cada linguagem regular está associada com um AFD mínimo.

Este resultado da existência de um AFD mínimo recebe o nome de **Teorema Myhill-Nerode**, em homenagem aos matemáticos John Myhill⁸ (1923-1987) e Anil Nerode (1932-), que o provaram na Universidade de Chicago em 1958 no artigo [49], de forma geral tal resultado fornece as condições suficientes e necessárias para que uma linguagem L seja regular, para construir tal resultado antes é necessário considerar algumas definições básicas e alguns resultados auxiliares.

Definição 6.16 (A família \mathcal{H}_L)

Seja L uma linguagem qualquer^a sobre o alfabeto Σ , para qualquer palavra w é definido o conjunto $L_w = \{x \mid wx \in L\}$. A família $\{L_w \mid w \in \Sigma^*\}$ construída sobre L será denotada por \mathcal{H}_L , ou seja, $\mathcal{H}_L = \{L_w \mid w \in \Sigma^*\}$

^aNão necessariamente regular.

Com respeito aos conjuntos L_w o leitor mais atento pode notar que $L_\lambda = L$, além disso, os conjuntos L_w também apresentam a seguinte propriedade básica.

Proposição 6.1

Dado $L \subseteq \Sigma^*$. Se $L_w = L_{w'}$, então $L_{wa} = L_{w'a}$ para todo $a \in \Sigma$.

Demonstração Suponha que $L_w = L_{w'}$, assim para todo $a \in \Sigma^*$ tem-se que:

$$\begin{aligned} x \in L_{wa} &\iff wax \in L \\ &\iff ax \in L_w \\ &\stackrel{Hip.}{\iff} ax \in L_{w'} \\ &\iff x \in L_{w'a} \end{aligned}$$

concluindo a prova. ■

Um fato importante sobre AFD que será usado a seguir e que não foi mencionado diretamente até agora é o exposto pelo resultado a seguir.

Proposição 6.2

Se $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ é um AFD, então $\widehat{\delta}(q_0, uv) = \widehat{\delta}(\widehat{\delta}(q_0, u), v)$ para todo $u, v \in \Sigma^*$. ♠

Demonstração A prova é por indução sobre o tamanho da palavra uv e ficará como exercício ao leitor. ■

O lema a seguir mostra que $\#\mathcal{H}_L$ é na verdade um limite inferior para o número de estados em um AFD.

Lema 6.1

Se $L = \mathcal{L}(A)$ para algum AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, então $\#\mathcal{H}_L \leq |Q|$. ♥

Demonstração Suponha que $L = \mathcal{L}(A)$ para algum AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, agora para todo $q \in Q$ defina um novo AFD A_q igual ao anterior em todos os aspectos menos no estado inicial pois este será o estado q , ou

⁸O professor Myhill também é conhecido por seu Teorema de isomorfismo[48], que pode ser visto como um análogo dentro da teoria da computabilidade ao teorema de Cantor–Bernstein–Schroeder e pelo famoso pelo Teorema de Rice–Myhill–Shapiro, mais comumente conhecido como Teorema de Rice [12, 53].

seja, $A_q = \langle Q, \Sigma, \delta, q, F \rangle$. Agora para toda palavra $w \in \Sigma^*$ suponha que $\widehat{\delta}(q_0, w) = q$, por definição note que,

$$\begin{aligned}
 x \in L_w & \stackrel{\text{Def. 6.16}}{\iff} wx \in L \\
 & \iff wx \in \mathcal{L}(A) \\
 & \iff \widehat{\delta}(q_0, wx) \in F \\
 & \stackrel{\text{Prop. 6.2}}{\iff} \widehat{\delta}(\widehat{\delta}(q_0, w), x) \in F \\
 & \stackrel{\text{Hip.}}{\iff} \widehat{\delta}(q, x) \in F \\
 & \iff x \in \mathcal{L}(A_q)
 \end{aligned}$$

Dessa forma tem-se que $\mathcal{L}(A_q) = L_w$, e obviamente $\mathcal{L}(A_{q_0}) = L$. Desde que A é fixo, tem-se que L_w depende apenas do estado obtido quando a computação w começa em q_0 , e assim o número de L_w distintos, ou seja, os elementos de \mathcal{H}_L não pode ser maior que o número de estados em A , portanto, $\#\mathcal{H}_L \leq |Q|$. ■

O próximo lema estabelece que para alguma linguagem L no caso \mathcal{H}_L ser finito, então sua cardinalidade será o limite superior no número de estados em um AFD capaz de reconhecer L .

Lema 6.2

Seja $L \subseteq \Sigma^*$. Se \mathcal{H}_L é finito, então existe um AFD A_L tal que $L = \mathcal{L}(A_L)$ e A_L possui exatamente $\#\mathcal{H}_L$ estados. ♡

Demonstração Dado $L \subseteq \Sigma^*$ assumo que \mathcal{H}_L é finito, dito isto pode-se construir o seguinte AFD $A_L = \langle \mathcal{H}_L, \Sigma, \delta, q_0, F \rangle$ onde:

$$q_0 = L \tag{6.28}$$

e para todo $L_w \in \mathcal{H}_L$ e $a \in \Sigma$ tem-se

$$\delta(L_w, a) = L_{wa} \tag{6.29}$$

e

$$F = \{L_w \in \mathcal{H}_L \mid \lambda \in L_w\} \tag{6.30}$$

sobre a definição de A_L por indução sobre o tamanho de $w \in \Sigma^*$ pode-se facilmente verificar que,

$$\widehat{\delta}(q_0, w) = L_w \tag{6.31}$$

além disso, claramente A_L possui exatamente $\#\mathcal{H}_L$ estados, dito isto, note que:

$$\begin{aligned}
 w \in L & \stackrel{\text{Def. 6.16}}{\iff} \lambda \in L_w \\
 & \stackrel{\text{Eq. (6.30)}}{\iff} L_w \in F \\
 & \stackrel{\text{Eq. (6.31)}}{\iff} \widehat{\delta}(q_0, w) \in F \\
 & \iff w \in \mathcal{L}(A_L)
 \end{aligned}$$

e portanto, $L = \mathcal{L}(A_L)$ concluindo assim a prova. ■

Definição 6.17 (Dimensão de AFD)

Seja $A = \langle Q, \Sigma, \delta', q_0, F \rangle$ um AFD a dimensão de A , denotado por $\dim(A)$, é igual a quantidade de estados em A , isto é, $\dim(A) = \#Q$. ♣

Definição 6.18 (AFD Mínimo)

Seja L uma linguagem regular tal que $L = \mathcal{L}(A)$ para algum um AFD A . O AFD A será dito ser mínimo se, e somente se, e para todo outro AFD B tal que $L = \mathcal{L}(B)$ tem-se que $\dim(A) \leq \dim(B)$.

O próximo resultado mostra que não existe nenhum autômato com menos estados que o AFD construído na prova do Lema 6.2, ou seja, o método de construção mostrado na na prova do Lema 6.2 gera o AFD mínimo de qualquer linguagem.

Lema 6.3

Seja L uma linguagem regular, assim o AFD A_L construído no Lema 6.2 é o único (a menos de isomorfismo) mínimo AFD que aceita L .

Demonstração Suponha que existe outro AFD mínimo $N = \langle Q, \Sigma, \delta', q'_0, F' \rangle$ tal que $L = \mathcal{L}(N)$ diferente do AFD $A_L = \langle \mathcal{H}_L, \Sigma, \delta, L, F \rangle$ construído na prova do Lema 6.2. Agora será definida uma função $f : Q \rightarrow \mathcal{H}_L$ definida simplesmente como:

$$f(q) = \begin{cases} L, & \text{se } q = q_0 \\ \mathcal{L}(A_q), & \text{senão} \end{cases}$$

para todo $q \in Q$, onde $\mathcal{L}(A_q)$ é da mesma forma que na prova do Lema 6.1 onde o AFD fixo é exatamente A_L . Por esta definição é claro que:

- (1) Se $q_i \neq q_j$, então tem-se que $f(q_i) \neq f(q_j)$, conseqüentemente a função f é injetora.
- (2) f preserva a condição de estado inicial.
- (3) Para $a \in \Sigma, w \in \Sigma^*$ e $q, p \in Q$ assumamos que $\delta(q, a) = p$ e $f(q) = \mathcal{L}(A_q) = L_w$ assim para qualquer $u \in \Sigma^*$ tem-se que

$$\begin{aligned} u \in f(p) &\iff u \in \mathcal{L}(A_p) \\ &\iff \widehat{\delta}(p, u) \in F \\ &\iff \widehat{\delta}(\widehat{\delta}(q, a), u) \in F \\ &\iff \widehat{\delta}(q, au) \in F \\ &\iff au \in \mathcal{L}(A_q) \\ &\iff au \in f(q) \\ &\iff au \in L_w \\ &\iff wau \in L \\ &\iff u \in L_{wa} \end{aligned}$$

portanto, $f(p) = L_{wa}$, ou seja, f preserva transições⁹.

- (4) Agora note que pela definição de estado final e pela construção de A_L tem-se que

$$q \in F' \iff \widehat{\delta}(q, \lambda) \in F' \iff \lambda \in \mathcal{L}(A_q) \iff \mathcal{L}(A_q) \in F \iff f(q) \in F$$

portanto, a função f preserva estados finais.

- (5) Agora dado $w \in \Sigma^*$ assumamos que $\widehat{\delta}(q_0, w) = q$, assim pela prova do Lema 6.1 para todo $L_w \in \mathcal{H}_L$, ou seja, para todo $L_w \in \text{Ima}(f)$ tem-se que $L_w = \mathcal{L}(A_q)$, e portanto, pela definição de f tem-se que existe $q \in \text{Dom}(f)$ tal que $L_w = f(q)$, ou seja, f é sobrejetora.

Desde que f é injetora e sobrejetora tem-se que f é uma bijeção, e portanto, $\#Q = \#\mathcal{H}_L$, logo $\dim(N) =$

⁹Isto é o mesmo que dizer que $f(\delta'(q, a)) = \delta(f(q), a)$.

$\dim(A_L)$. Agora desde que f preserva o estado inicial, os estados finais e as transições tem-se que f é um isomorfismo do AFN N para o AFD A_L , e portanto, eles são o mesmo AFD se diferenciando apenas pela rotulação dos seus estados. ■

Pode-se agora finalmente enunciar o Teorema Myhill-Nerode que estabelece uma caracterização para as linguagens regulares.

Teorema 6.5 (Teorema Myhill-Nerode)

Uma linguagem $L \subseteq \Sigma^*$ será regular se, e somente se, \mathcal{H}_L é finito e existe um AFD mínimo A com exatamente $\#\mathcal{H}_L$ estados tal que $\mathcal{L}(A) = L$. ♡

Demonstração Direto dos Lemas 6.1, 6.2 e 6.3. ■

Apesar de extremamente elegante o resultado exposto pelo Teorema Myhill-Nerode apresentado anteriormente, não fornece um procedimento (ou algoritmo) geral explícito para construir um AFD mínimo equivalente a outro AFD dado. Esse processo de obter um AFD mínimo é chamado de minimização [12], e o mesmo é baseado na ideia de relação equivalência entre estados e do espaço quociente de estados em um AFD. Primeiro será apresentado a formalização matemática da construção do AFD quociente e depois o pseudo-código de tal construção.

Definição 6.19 (Estados Equivalentes)

Seja $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ um AFD. A relação de equivalência entre dois estados $q, q' \in Q$ será denotado por $q \equiv q'$ e será verdadeira quando para todo $w \in \Sigma^*$ tem-se que $\widehat{\delta}(q, w) \in F \iff \widehat{\delta}(q', w) \in F$. ♣

O resultado a seguir diz que se dois estados são equivalentes, então seus sucessores também o são.

Lema 6.4

Dado um AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$. Se $q \equiv q'$, então $\delta(q, a) \equiv \delta(q', a)$. ♡

Demonstração Dado $a \in \Sigma$, por contrapositiva será mostrado que:

$$\text{Se } \delta(q, a) \not\equiv \delta(q', a), \text{ então } q \not\equiv q'.$$

Inicialmente assuma que $\delta(q, a) \not\equiv \delta(q', a)$, assim pela Definição 6.19 existe um $w \in \Sigma^*$ tal que um dos dois casos a seguir acontece:

- (1) $\widehat{\delta}(\delta(q, a), w) \in F$ e $\widehat{\delta}(\delta(q', a), w) \notin F$ ou
- (2) $\widehat{\delta}(\delta(q, a), w) \notin F$ e $\widehat{\delta}(\delta(q', a), w) \in F$.

em particular quando $w = \lambda$, para o primeiro caso (a prova é similar para o caso (2)) tem-se pela Definição 6.2 que:

$$\widehat{\delta}(\delta(q, a), w) \in F \text{ e } \widehat{\delta}(\delta(q', a), w) \notin F \iff \delta(q, a) \in F \text{ e } \delta(q', a) \notin F$$

e desde que $a \in \Sigma^*$ pela Definição 6.19 tem-se que $q \not\equiv q'$, o que conclui a prova da contrapositiva. Desde que a contrapositiva é verdadeira tem-se que afirmação original “Se $q \equiv q'$, então $\delta(q, a) \equiv \delta(q', a)$ ” é também verdadeira. ■

Usando a definição anterior, a seguir será apresentado a definição de AFD (ou colapso) quociente sobre um determinado AFD dado.

Definição 6.20 (AFD quociente)

Seja $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ um AFD. O AFD (ou colapso) quociente de A é o AFD $A_{/\equiv} = \langle Q_{/\equiv}, \Sigma, \delta^*, [q_0], F_{/\equiv} \rangle$ e para todo $q \in Q$ e $a \in \Sigma$ tem-se que $\delta^*([q], a) = [\delta(q, a)]$ e $F_{/\equiv} = \{[q] \mid q \in F\}$.

Teorema 6.6

Seja $A_{/\equiv} = \langle Q_{/\equiv}, \Sigma, \delta^*, [q_0], F_{/\equiv} \rangle$ o AFD quociente obtido a partir de um AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$. Então para todo $w \in \Sigma^*$ tem-se que $\widehat{\delta^*}([q], w) = [\widehat{\delta}(q, w)]$.

Demonstração Por indução sobre o tamanho de $w \in \Sigma^*$ será mostrado a seguinte igualdade $\widehat{\delta^*}([q], w) = [\widehat{\delta}(q, w)]$.

- **Base da indução:**

Quando $|w| = 0$ ($w = \lambda$) tem-se trivialmente que $\widehat{\delta^*}([q], \lambda) = [q] = [\widehat{\delta}(q, \lambda)]$.

- **Hipótese indutiva (HI):**

Suponha que para todo $w \in \Sigma^*$ com $|w| \geq 0$ tem-se que $\widehat{\delta^*}([q], w) = [\widehat{\delta}(q, w)]$

- **Passo indutivo:**

Dado $w = ua$ com $u \in \Sigma^*$, $|u| \geq 0$ e $a \in \Sigma$ tem-se que,

$$\begin{aligned}
 \widehat{\delta^*}([q], w) &= \widehat{\delta^*}([q], ua) \\
 &= \delta^*(\widehat{\delta^*}([q], u), a) \\
 &\stackrel{(HI)}{=} \delta^*([\widehat{\delta}(q, u)], a) \\
 &= [\delta(\widehat{\delta}(q, u), a)] \\
 &= [\widehat{\delta}(q, ua)] \\
 &= [\widehat{\delta}(q, w)]
 \end{aligned}$$

O que completa a prova. ■

Teorema 6.7

Seja $A_{/\equiv} = \langle Q_{/\equiv}, \Sigma, \delta^*, [q_0], F_{/\equiv} \rangle$ o AFD quociente obtido a partir de um AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$. Então $q \in F \iff [q] \in F_{/\equiv}$

Demonstração (\implies) Trivial pela própria definição de $F_{/\equiv}$. (\impliedby) É suficiente mostrar que se $q \equiv p$ e $q \in F$, então $p \in F$. Para provar isto, suponha que $q \equiv p$ e $q \in F$, mas note que $q \in F \iff \widehat{\delta}(q, \lambda) \in F$, mas como $q \equiv p$, por definição tem-se para todo $w \in \Sigma^*$ que $\widehat{\delta}(q, w) \in F \iff \widehat{\delta}(p, w) \in F$, assim no particular quando $w = \lambda$ tem-se que $\widehat{\delta}(p, \lambda) \in F$, mas $\widehat{\delta}(p, \lambda) = p$, portanto, $p \in F$. ■

O próximo resultado mostra que o colapso de um AFD para seu AFD quociente preserva a linguagem aceita pelo autômato.

Teorema 6.8

Seja $A_{/\equiv} = \langle Q_{/\equiv}, \Sigma, \delta^*, [q_0], F_{/\equiv} \rangle$ o AFD quociente obtido a partir de um AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$. Então $\mathcal{L}(A_{/\equiv}) = \mathcal{L}(A)$.

Demonstração Basta notar que para qualquer $w \in \Sigma^*$ tem-se que:

$$w \in \mathcal{L}(A_{/\equiv}) \iff \widehat{\delta^*}([q_0], w) \in F_{/\equiv} \stackrel{Teo. 6.6}{\iff} [\widehat{\delta}(q_0, w)] \in F_{/\equiv} \stackrel{Teo. 6.7}{\iff} \widehat{\delta}(q_0, w) \in F \iff w \in \mathcal{L}(A)$$

logo $\mathcal{L}(A_{/\equiv}) = \mathcal{L}(A)$. ■

Algoritmo 3: Algoritmo para construção do AFD quociente.

Entrada: Um AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$
Saída: O AFD quociente $A_{/\equiv} = \langle Q_{/\equiv}, \Sigma, \delta^*, [q_0], F_{/\equiv} \rangle$

```

1 início
2   Defina uma matriz  $M$  de dimensão  $\#Q$  por  $\#Q$ 
3   para cada  $(i, j) \in \{0, \dots, \#Q - 1\} \times \{0, \dots, \#Q - 1\}$  tal que  $j > i$  faça
4     se  $q_i \in F, q_j \notin F$  ou  $q_i \notin F, q_j \in F$  então
5       Marque a posição  $M(i, j)$  com um  $X$ 
6     fim
7   fim
8   para cada  $(i, j) \in \{0, \dots, \#Q - 1\} \times \{0, \dots, \#Q - 1\}$  tal que  $j > i$  faça
9     se  $M(i, j)$  não foi marcado então
10      para cada  $a \in \Sigma$  faça
11        se  $\delta(q_i, a) \in F, \delta(q_j, a) \notin F$  ou  $\delta(q_i, a) \notin F, \delta(q_j, a) \in F$  então
12          Marque a posição  $M(i, j)$  com um  $X$ 
13        fim
14      fim
15    fim
16  fim
17  Defina  $T = \{0, \dots, \#Q\}$ 
18  Defina  $Q_{/\equiv}$  como sendo vazio
19  enquanto  $T \neq \emptyset$  faça
20    Selecione o menor  $i \in T$ 
21    se  $\nexists [q] \in Q_{/\equiv}$  tal que  $q_i \in [q]$  então
22      Defina uma classe de equivalência  $[q_i] = \{q_i\}$ 
23      para todo  $j \in T$  tal que  $i \neq j$  faça
24        se  $M(i, j)$  não estiver marcado então
25          Adicione  $q_j$  na classe de equivalência  $[q_i]$ 
26          Remova  $j$  de  $T$ 
27        fim
28      fim
29    fim
30    Remova  $i$  de  $T$ 
31  fim
32  para cada  $([q], a) \in Q_{/\equiv} \times \Sigma$  faça
33    Defina  $\delta^*([q], a) = [\delta(q, a)]$ 
34  fim
35  Considere inicialmente  $F_{/\equiv}$  como vazio
36  para cada  $[q] \in Q_{/\equiv}$  faça
37    se  $q \in F$  então
38      Adicione  $[q]$  ao conjunto  $F_{/\equiv}$ 
39    fim
40  fim
41  retorna  $A_{/\equiv} = \langle Q_{/\equiv}, \Sigma, \delta^*, [q_0], F_{/\equiv} \rangle$ 
42 fim

```

Agora é aceitável que o leitor possa se questionar sobre o que acontece quando se tenta colapsar um AFD que já é um AFD quociente. A resposta para esse fato é simples, não acontece nada, isto é, se $A_{/\equiv}$ é o AFD

quociente, então o colapso dele é ele próprio, a prova disto pode ser vista em [41].

Observação: Um fato importante sobre o AFD quociente é que ele é mínimo a menos de isomorfismo [41], isto é, a construção do quociente é equivalente ao Teorema Myhill-Nerode.

Agora obviamente a construção do AFD quociente pode ser vista como um algoritmo, nesse sentido o algoritmo consiste em realizar três tarefas básicas, a primeira é de encontrar os estados do autômato que são equivalentes, depois o algoritmo deve construir as classes de equivalência dos estados em reunir todas as classes no conjunto (ou espaço) quociente Q/\equiv , depois disso o algoritmo deve definir as transições entre os estados são agora representados pelas classes de equivalência, isto é, o algoritmo deve construir a função δ^* , e por fim, definir um conjunto de estados finais, onde novamente os estados são classes de equivalência, ou seja, o último passo do algoritmo é construir o conjunto F/\equiv .

Em termos de “implementação” o Algoritmo 3 a seguir esboça o pseudo-código para a construção do AFD quociente, a ideia do mesmo é, primeiro usar os elementos em uma matriz de relação para marcar estados equivalentes, depois a partir desta matriz construir as classes de equivalência. Obviamente este texto não está preocupado com desempenho, assim possivelmente o pseudo-código a seguir pode não ser o mais eficiente, em termos da análise de algoritmo.

Observação: Um ponto a ressaltar é que o Algoritmo 3 não utilizar a matriz inteira para representar a relação de equivalência, isso acontece por que toda relação de equivalência é reflexiva e simétrica, assim basta se preocupar com as posições acima da diagonal principal da matriz, pois como explicado em [12], os elementos abaixo da diagonal principal são na verdade a contraparte simétrica da relação de equivalência e os elementos na diagonal são as informações da parte reflexiva da relação.

Exemplo 6.27 Inicialmente considere que o Algoritmo 3 recebe como entrada o AFD representado pelo grafo de transições esboçado pela Figura 6.13 a seguir.

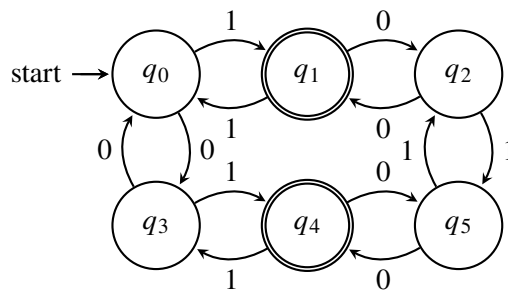


Figura 6.13: Grafo do AFD do Exemplo 6.24.

O algoritmo gera a matriz representada na Tabela 6.3 a seguir, em que a i -ésima linha (coluna) representa o estado q_{i-1} , e os espaços marcados com – são aqueles não considerados pelo algoritmo, as posições com X representam os estados não equivalentes, e as posições (i, j) em branco representam os estados equivalentes.

-	X	X		X	X
-	-	X	X		X
-	-	-	X	X	
-	-	-	-	X	X
-	-	-	-	-	X
-	-	-	-	X	-

Tabela 6.3: Matriz de equivalência M para o AFD 6.13.

A partir desta Tabela 6.3 o Algoritmo 3 gera as seguintes classes de equivalência: $[q_0] = \{q_0, q_3\}$, $[q_1] = \{q_1, q_4\}$ e $[q_2] = \{q_2, q_5\}$, depois define as transições e o conjunto de estados finais como esboçadas pelo grafo de Transição na Figura 6.14.

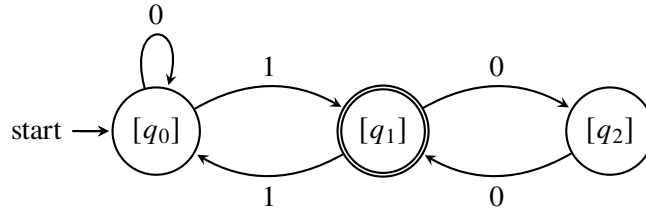


Figura 6.14: Grafo do AFD do Exemplo 6.24.

Desde que $\delta^*([q], a) = [\delta(q, a)]$ é um caso particular do resultado obtido no Teorema 6.6, e pela condição suficiente e necessária apresentada no Teorema 6.7, tem-se que a corretude do Algoritmo 3 se resume a demonstrar o teorema a seguir.

Teorema 6.9 (Corretude do Algoritmo de Construção do Quociente)

Seja $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ um AFD. Considerando o Algoritmo 3 e os índices $j > i$ tem-se que, $M(i, j)$ está marcado com X se, e somente se, $q_i \neq q_j$.

Demonstração A demonstração fica como exercício ao leitor. ■

6.5 Máquinas de Mealy e Moore

Os trabalhos iniciais de Mealy [44] e Moore [47], além de serem alicerces fundamentais da teoria dos autômatos finitos, também são os trabalhos responsáveis por formalizar o conceito de tradutores finitos ou autômato finito com saída. Diferente dos autômatos sem saída vistos nas seções 6.1, 6.2 e 6.3 as máquinas de Mealy e Moore são capazes de apresentar saídas mais complexas que as típicas saídas booleanas (aceita/rejeita) dos AFD, AFN e λ -AFN. As máquinas de Mealy, por exemplo, são um modelo matemático simples para definir máquinas de cifras (criptografia), ou seja, máquinas que a partir de um texto de entrada geram um texto cifrado (criptografado) como saída [45].

Em termos informais uma autômato com saída pode ser descrito como um máquina que possui duas memórias (somente de leitura), a primeira memória guarda a palavra de entrada usada no funcionamento da máquina, já segunda memória (apenas de escrita) é responsável por armazenar a palavra de saída, sendo essa gerada durante o funcionamento do autômato.

Definição 6.21 (Máquina de Mealy)

Uma máquina de Mealy é uma estrutura $A = \langle Q, \Sigma, \Lambda, \delta, \psi, q_0, \rangle$ onde Q, Σ, δ e q_0 são da mesma forma que na Definição 6.1. Λ é um alfabeto chamado **alfabeto de saída** e $\psi : Q \times \Sigma \rightarrow \Lambda^+$ é a função de tradução da máquina de Mealy.

Exemplo 6.28 A estrutura $M = \langle \{q_0, q_1\}, \{a, b\}, \{0, 1, 2\}, \delta, \psi, q_0 \rangle$ onde a função de transição é representada pela Tabela 6.4 e a função de tradução é representada pela Tabela 6.5 a seguir é uma máquina de Mealy.

$Q \backslash \Sigma$	Σ	
	a	b
q_0	q_1	q_1
q_1	q_0	q_1

Tabela 6.4: Tabela da Função de transição da Máquina de Mealy do Exemplo 6.28.

$Q \backslash \Sigma$	a	b
q_0	11	20
q_1	12	02

Tabela 6.5: Tabela da Função de tradução da Máquina de Mealy do Exemplo 6.28.

Pode-se utilizar a ideia de grafo de transição para representar visualmente as máquinas de Mealy a única diferença que esta representação terá em relação aos grafos que representam os AFD, é que as arestas que ligam dois vértices q_i e q_j serão rotuladas com pares $(a, x) \in \Sigma \times \Lambda$ representando assim simultaneamente as funções $\delta(q_i, a) = q_j$ e $\psi(q_i, a) = x$.

Exemplo 6.29 A representação do grafo de transição da máquina de Mealy do Exemplo 6.28 é como se segue.

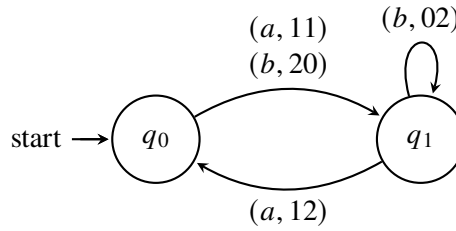


Figura 6.15: Representação da máquina de Mealy do Exemplo 6.28.

Note que a máquina de Mealy traduz (ou associa) cada transição da máquina a um símbolo do alfabeto de saída, ou seja, cada movimento no processo de computação da máquina é traduzido na saída da mesma. Obviamente $\hat{\delta}$ é definido da mesma forma que para os AFD, sendo assim basta agora estender a função ψ para que a máquina de Mealy possa de fato ser vista como um tradutor finito, e isto será feito a seguir.

Definição 6.22 (Extensão da função ψ)

Seja $A = \langle Q, \Sigma, \Lambda, \delta, \psi, q_0 \rangle$ uma máquina de Mealy a função ψ é estendida para a função $\hat{\psi} : Q \times \Sigma^* \rightarrow \Lambda^*$ usando recursividade como se segue.

$$\hat{\psi}(q, \lambda) = \lambda \quad (6.32)$$

$$\hat{\psi}(q, wa) = \hat{\psi}(q, w)\psi(\hat{\delta}(q, w), a) \quad (6.33)$$

Exemplo 6.30 Considerando a máquina de Mealy do Exemplo 6.28 a tradução da palavra bab a partir do estado q_1 é dada seguinte forma,

$$\begin{aligned}
 \hat{\psi}(q_1, bab) &= \hat{\psi}(q_1, ba)\psi(\hat{\delta}(q_1, ba), b) \\
 &= \hat{\psi}(q_1, b)\psi(\hat{\delta}(q_1, b), a)\psi(\hat{\delta}(q_1, ba), b) \\
 &= \hat{\psi}(q_1, \lambda)\psi(\hat{\delta}(q_1, \lambda), b)\psi(\hat{\delta}(q_1, b), a)\psi(\hat{\delta}(q_1, ba), b) \\
 &= \lambda 021220 \\
 &= 021220
 \end{aligned}$$

Observação: Pode-se interpretar que as funções ψ e $\hat{\psi}$ escrevem na fita de saída da máquina, isto é, na memória adicional de escrita que tais máquinas possuem.

Diferentemente dos AFD, AFN e λ -AFN o conjunto saída de qualquer máquina de Mealy não visto como uma linguagem aceita pela máquina, em vez disso, ela é vista como a linguagem traduzida pela máquina.

Definição 6.23 (Linguagem Traduzida - Máquina de Mealy)

Dado uma máquina de Mealy $A = \langle Q, \Sigma, \Lambda, \delta, \psi, q_0 \rangle$ a linguagem traduzida de A , denotado por $TRAD(A)$, é o conjunto de todas as traduções das palavras $w \in \Sigma^*$ realizadas a partir do estado inicial q_0 , ou seja, tem-se que:

$$TRAD(A) = \{\widehat{\psi}(q_0, w) \mid w \in \Sigma^*\} \quad (6.34)$$

Em contrapartida as máquinas de Mealy, que como já dito, traduzem cada transição da máquina em um fragmento da palavra de saída, existem as máquina de Moore, nomeadas em homenagem a Edward F. Moore (1925-2003) que as propôs em seu seminal artigo [47], tais máquinas não traduzem as transições em prefixos da palavras de saída, em vez disso, cada estado da máquina é mapeado em um símbolo do alfabeto de saída.

Definição 6.24 (Máquina de Moore)

Uma máquina de Moore é uma estrutura $A = \langle Q, \Sigma, \Lambda, \delta, \Phi, q_0 \rangle$ onde Q, Σ, δ e q_0 são da mesma forma que na Definição 6.1. Λ é um alfabeto chamado **alfabeto de saída** e $\Phi : Q \rightarrow \Lambda^+$ é a função de escrita dos estado da máquina de Moore.

Exemplo 6.31 A estrutura $M = \langle \{q_0, q_1, q_2\}, \{a, b\}, \{x, y, z\}, \delta, \Phi, q_0 \rangle$ onde a função de transição é esboçada pela Tabela 6.6 a seguir, sendo a função de escrita dos estados é definida como: $\Phi(q_0) = xy$, $\Phi(q_1) = yz$ e $\Phi(q_2) = zx$. É uma máquina de Moore.

$Q \backslash \Sigma$	a	b
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_0	q_2

Tabela 6.6: Tabela de transição para a máquina de Moore do Exemplo 6.31.

Similar aos AFD pode-se utilizar a ideia de grafo de transição para representar visualmente as máquinas de Moore a única diferença que esta representação terá em relação aos grafos que representam os AFD, é que os vértices serão rotulados por pares $(q, w) \in Q \times \Lambda$ e q_j em que $\Phi(q) = w$.

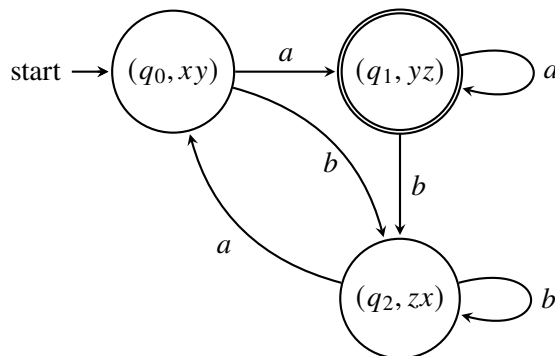


Figura 6.16: Representação da máquina de Moore do Exemplo 6.28.

Exemplo 6.32 A representação do grafo de transição da máquina de Mealy do Exemplo 6.31 pode ser observada na Figura 6.16.

Observação: Note que para qualquer $w \in \Sigma^*$ tem-se que $\Phi(\widehat{\delta}(q_0, w))$ expressa a saída da máquina de Moore para a palavra w .

Para que as máquinas de Moore possam ser vistas como tradutores é necessário apresentar uma função de tradução para tais máquinas, sendo esta apresentada a seguir.

Definição 6.25 (Função de tradução das máquinas de Moore)

Dado uma máquina de Moore $A = \langle Q, \Sigma, \Lambda, \delta, \Phi, q_0 \rangle$ a função de tradução $\widehat{\Phi} : Q \times \Sigma^* \rightarrow \Lambda^*$ é definida por recursão como se segue.

$$\widehat{\Phi}(q, \lambda) = \lambda \quad (6.35)$$

$$\widehat{\Phi}(q, wa) = \widehat{\Phi}(\delta(q, w))\Phi(\delta(q, wa)) \quad (6.36)$$

Observação: Pode-se assim como para o caso das máquinas de Mealy considerar que a função de tradução das máquinas de Moore, isto é, a função $\widehat{\Phi}$ escreve a memória de escrita da máquina.

Definição 6.26 (Linguagem Traduzida - Máquina de Moore)

Dado uma máquina de Moore $A = \langle Q, \Sigma, \Lambda, \delta, \Phi, q_0 \rangle$ a linguagem traduzida de A , denotado por $TRAD(A)$, é o conjunto de todas as traduções das palavras $w \in \Sigma^*$ realizadas a partir do estado inicial q_0 , ou seja, tem-se que:

$$TRAD(A) = \{\widehat{\Phi}(q_0, w) \mid w \in \Sigma^*\} \quad (6.37)$$

O leitor atento deve ter notado pelas definições que as traduções feitas pelas máquinas de Mealy e Moore, apresentam a propriedade de involução sobre a palavra vazia. Os próximos resultados apresentam uma forte relação entre as máquinas de Mealy e Moore.

Teorema 6.10 (Transformação Moore-Mealy)

Dado uma linguagem $L \subseteq \Lambda^*$. Se $L = TRAD(N)$ para alguma máquina de Moore N , então existe uma máquina de Mealy M tal que $L = TRAD(M)$.

Demonstração Suponha que $L = TRAD(N)$ para alguma máquina de Moore $N = \langle Q, \Sigma, \Lambda, \delta, \Phi, q_0 \rangle$, agora defina uma máquina de Mealy $M = \langle Q, \Sigma, \Lambda, \delta, \psi, q_0 \rangle$, onde para todo $(q, a) \in Q \times \Sigma$ é definido que:

$$\psi(q, a) = \Phi(\delta(q, a)) \quad (6.38)$$

Agora será mostrado por indução sobre o tamanho das palavras que para todo $w \in \Sigma^+$ que as traduções de M e N são iguais, ou seja, será mostrado que $\widehat{\Phi}(q_0, w) = \widehat{\psi}(q_0, w)$.

- **Base da indução:** Quando $|w| = 0$ é trivial da própria definição de $\widehat{\Phi}$ e $\widehat{\psi}$.
- **Hipótese indutiva (HI):** Suponha que para todo $|w| = n$ com $n \geq 0$ tem-se que $\widehat{\Phi}(q_0, w) = \widehat{\psi}(q_0, w)$.
- **Passo indutivo:** Dado $|w| = ua$ com $u \in \Sigma^*$ tal que $|u| \geq 0$ e $a \in \Sigma$ tem-se que,

$$\begin{aligned} \widehat{\Phi}(q_0, w) &= \widehat{\Phi}(q_0, ua) \\ &= \widehat{\Phi}(q_0, u)\Phi(\delta(q_0, ua)) \\ &\stackrel{(HI)}{=} \widehat{\psi}(q_0, u)\Phi(\delta(q_0, ua)) \\ &= \widehat{\psi}(q_0, u)\Phi(\delta(\delta(q_0, u), a)) \\ &\stackrel{Eq. 6.38}{=} \widehat{\psi}(q_0, u)\psi(\delta(q_0, u), a) \\ &= \widehat{\psi}(q_0, ua) \\ &= \widehat{\psi}(q_0, w) \end{aligned}$$

Consequentemente, $TRAD(N) = TRAD(M)$ e, portanto, $L = TRAD(M)$. ■

Teorema 6.11 (Transformação Mealy-Moore)

Dado uma linguagem $L \subseteq \Lambda^$. Se $L = TRAD(N)$ para alguma máquina de Mealy N , então existe uma máquina de Moore M tal que $L = TRAD(M)$.* ♥

Demonstração Similar ao raciocínio da demonstração do Teorema 6.10, e ficará como exercício ao leitor. ■

Observação: As definições dadas aqui não apresentam o conceito de estado final, para leitores interessados em máquinas de Mealy e Moore com estados finais ver [45].

6.6 A Notação Matricial

Escrever depois...

6.7 Expressões Regulares

Até agora as linguagens foram vistas sobre a ótica das máquinas de computação, isto é, sobre a perspectiva dos autômatos finitos, em tal perspectiva as linguagens são vistas como sendo conjuntos de palavras sobre os quais as máquinas tinha a tarefa de reconhecer seus elementos.

Neste seção será apresentada uma nova visão de aspecto mais algébrico para as linguagens, essa nova visão foi introduzida por Kleene em seu seminal *paper* “Representation of events in nerve nets and finite automata” [34], tal perspectiva consiste em um sistema formal (com sintaxe e semântica) chamado de expressões regulares, a seguir é formalizado a sintaxe das expressões regulares.

Definição 6.27 (Conjunto das Expressões Regulares – Sintaxe)

Seja Σ uma alfabeto, o conjunto de todas as expressões regulares sobre Σ , denotado por Exp_{Σ} , é o conjunto indutivamente gerado pelas seguintes regras.

(B)ase: \emptyset, λ e cada $a \in \Sigma$, são expressões regulares^a.

(P)asso indutivo: Se $r_1, r_2 \in Exp_{\Sigma}$, então $r_1 + r_2, r_1 \cdot r_2, r_1^*, (r_1) \in Exp_{\Sigma}$.

(F)echo: Exp_{Σ} é exatamente o conjunto dos elementos obtidos a partir **(B)** ou usando-se uma quantidade finita (podendo ser nula) de aplicações de **(P)**.

^aAs expressões regulares da base costumam ser chamadas de expressões regulares primitivas. ♣

No que diz respeito as expressões regulares é comum assumir que $+, \cdot, (,), ^*, \notin \Sigma$, assim uma expressão regular nem sempre é uma palavra sobre Σ , em geral os símbolos $+$ e \cdot são lidos como soma e produto [17]. Além disso, como dito em [12] se $r_1, r_2 \in Exp_{\Sigma}$, então costuma-se escrever $r_1 r_2$ em vez de $r_1 \cdot r_2$.

Observação: Também é possível encontrar referência em que o termo expressão regular seja trocado para álgebra de Kleene.

Exemplo 6.33 Considerando o alfabeto $\{0, 1\}$ tem-se que $(1 + 1)0, 01 \in Exp_{\Sigma}$. Essa afirmação pode ser verificada construindo tais palavras facilmente, basta pela regra **(B)** tem-se que 0 e 1 são expressões regulares primitivas, assim usando a regra **(P)** pode-se construir as expressões $1 + 1$ e 01 , agora aplicando novamente a regra **(P)** sobre a expressão $1 + 1$ pode-se gerar a expressão $(1 + 1)$, finalmente, aplicando **(P)** novamente obtem-se a expressão $(1 + 1)0$ e isso mostra que de fato $(1 + 1)0, 01 \in Exp_{\Sigma}$.

Exemplo 6.34 Considerando o alfabeto $\{a, b, c\}$ tem-se que $a(\emptyset + (bc^*)) \in Exp_{\Sigma}$. Essa afirmação pode ser

verificada construindo tal palavra, uma vez que, a , b e c são expressões regulares primitivas aplicando o passo **(B)** varias vezes será obtida a expressão regular $a(\emptyset + (bc)^*)$.

A seguir será formalizado o conceito de semântica para as expressões regulares, sendo que tal semântica pode ser visto como uma **semântica denotacional**¹⁰ [55], que apresenta significado as operações de soma e multiplicação.

Definição 6.28 (Semântica das Expressão Regulares)

Seja Exp_{Σ} o conjunto das expressões regulares sobre Σ , a semântica (ou interpretação) de Exp_{Σ} é uma função $\mathcal{L} : Exp_{\Sigma} \rightarrow \wp(\Sigma^*)$ definida recursivamente para todo $r, r_1, r_2 \in Exp_{\Sigma}$ pelas seguintes regras.

- (i) Se $r \in \Sigma \cup \{\lambda\}$, então $\mathcal{L}(r) = \{r\}$.
- (ii) Se $r = \emptyset$, então $\mathcal{L}(r) = \emptyset$.
- (iii) Se $r = r_1 + r_2$, então $\mathcal{L}(r) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$.
- (iv) Se $r = r_1 \cdot r_2$, então $\mathcal{L}(r) = \mathcal{L}(r_1)\mathcal{L}(r_2)$.
- (v) Se $r = r_1^*$, então $\mathcal{L}(r) = (\mathcal{L}(r_1))^*$.
- (vi) Se $r = (r_1)$, então $\mathcal{L}(r) = (\mathcal{L}(r_1))$.



Agora como explicado em [17], para a valoração de expressões regulares não primitivas, é necessário que seja seguido a precedência dos operadores no momento de combinar as linguagens, sendo tal precedência no sentido de maior precedência para a menor formada pela seguinte ordem: fecho de Kleene, concatenação e união.

Observação: Vale destacar que como na aritmética convencional os parênteses mudam a precedência dos conectivos anteriores, e deve ser avaliados dos mais internos para os mais externos.



Nota: A valoração pode vir a gerar situações como (L) onde $L \subseteq \Sigma^*$, neste caso será escrito simplesmente L vez de (L) .

Exemplo 6.35 Dado o alfabeto $\{0, 1\}$ e $(00)^* \in Exp_{\Sigma}$ tem-se que:

$$\begin{aligned}
 \mathcal{L}((00)^*) &= (\mathcal{L}((00)))^* \\
 &= ((\mathcal{L}(00)))^* \\
 &= ((\mathcal{L}(0)\mathcal{L}(0)))^* \\
 &= ((\{0\}\{0\}))^* \\
 &= ((\{00\}))^* \\
 &= (\{00\})^* \\
 &= \{00\}^* \\
 &= \{\lambda, 00, 0000, 000000, 00000000, \dots\}
 \end{aligned}$$

ou seja, a valoração da expressão regular $(00)^*$ consiste da linguagem de todas as palavras w sobre o alfabeto

¹⁰Uma semântica denotacional é aquela em que as funções de valoração usadas, são funções que mapeiam palavras da linguagem para funções parciais que representam o comportamento dos programas.

$\{0, 1\}$ sem nenhum 1 e que o tamanho seja par, isto é, $|w| = 2k$ para algum $k \in \mathbb{N}$.

Exemplo 6.36 Dado o alfabeto $\{0, 1\}$ e $(0 + 1^*)0 \in \text{Exp}_\Sigma$ tem-se que:

$$\begin{aligned}
 \mathcal{L}((0 + 1^*)0) &= \mathcal{L}((0 + 1^*))\mathcal{L}(0) \\
 &= (\mathcal{L}(0 + 1^*))\{0\} \\
 &= (\mathcal{L}(0) \cup \mathcal{L}(1^*))\{0\} \\
 &= (\mathcal{L}(0) \cup (\mathcal{L}(1))^*)\{0\} \\
 &= (\{0\} \cup \{1\}^*)\{0\} \\
 &= (\{0\} \cup \{\lambda, 1, 11, 111, 1111, \dots\})\{0\} \\
 &= \{\lambda, 0, 1, 11, 111, 1111, \dots\}\{0\} \\
 &= \{0, 00, 10, 110, 1110, 11110, \dots\}
 \end{aligned}$$

ou seja, a valoração da expressão regular $\{0, 1\}$ e $(0 + 1^*)0$ é exatamente a linguagem de todas as palavras w sobre o alfabeto $\{0, 1\}$ sendo que $w = 0^m$ ou $w = 1^n0$ com $m, n \in \mathbb{N}$ tal que $1 \leq m \leq 2$ e $n \geq 1$.

Exemplo 6.37 Dado o alfabeto $\{a, b, c\}$ e a expressão $((ab)^* + c)\emptyset \in \text{Exp}_\Sigma$ tem-se que:

$$\begin{aligned}
 \mathcal{L}(((ab)^* + c)\emptyset) &= \mathcal{L}(((ab)^* + c))\mathcal{L}(\emptyset) \\
 &= (\mathcal{L}((ab)^* + c))\emptyset \\
 &= (\mathcal{L}((ab)^*) \cup \mathcal{L}(c))\emptyset \\
 &= ((\mathcal{L}((ab)))^* \cup \{c\})\emptyset \\
 &= (((\mathcal{L}(ab)))^* \cup \{c\})\emptyset \\
 &= (((\mathcal{L}(a)\mathcal{L}(b)))^* \cup \{c\})\emptyset \\
 &= (((\{a\}\{b\}))^* \cup \{c\})\emptyset \\
 &= (((\{ab\}))^* \cup \{c\})\emptyset \\
 &= ((\{ab\})^* \cup \{c\})\emptyset \\
 &= (\{ab\}^* \cup \{c\})\emptyset \\
 &= (\{\lambda, ab, abab, ababab, \dots\} \cup \{c\})\emptyset \\
 &= (\{\lambda, c, ab, abab, ababab, \dots\})\emptyset \\
 &= \{\lambda, c, ab, abab, ababab, \dots\}\emptyset \\
 &= \{\lambda, c, ab, abab, ababab, \dots\}
 \end{aligned}$$

portanto, a valoração da expressão regular $((ab)^* + c)\emptyset$ é exatamente a linguagem de todas as palavras w sobre o alfabeto $\{a, b, c\}$ onde $w = c^m$ ou $w = (ab)^n$ com $m, n \in \mathbb{N}$ tal que $m \leq 1$ e $n \geq 1$.

Definição 6.29

Duas expressões regulares $r_1, r_2 \in \text{Exp}_\Sigma$ são ditas equivalentes, denotado por $r_1 \equiv r_2$, sempre que $\mathcal{L}(r_1) = \mathcal{L}(r_2)$.



Exemplo 6.38 Dada a expressão $00 + (1^*0)$ note que:

$$\begin{aligned}
 \mathcal{L}(00 + (1^*0)) &= \mathcal{L}(00) \cup \mathcal{L}((1^*0)) \\
 &= \mathcal{L}(0)\mathcal{L}(0) \cup (\mathcal{L}(1^*0)) \\
 &= \mathcal{L}(0)\mathcal{L}(0) \cup (\mathcal{L}(1^*)\mathcal{L}(0)) \\
 &= \mathcal{L}(0)\mathcal{L}(0) \cup ((\mathcal{L}(1))^*\mathcal{L}(0)) \\
 &= \{0\}\{0\} \cup ((\{1\})^*\{0\}) \\
 &= \{0\}\{0\} \cup (\{1\}^*\{0\}) \\
 &= \{0\}\{0\} \cup (\{\lambda, 1, 11, 111, 1111, \dots\}\{0\}) \\
 &= \{0\}\{0\} \cup (\{0, 10, 110, 1110, 11110, \dots\}) \\
 &= \{0\}\{0\} \cup \{0, 10, 110, 1110, 11110, \dots\} \\
 &= \{00\} \cup \{0, 10, 110, 1110, 11110, \dots\} \\
 &= \{0, 00, 10, 110, 1110, 11110, \dots\}
 \end{aligned}$$

logo, tal expressão é equivalente a expressão $\equiv (0 + 1^*)0$ apresentada no Exemplo 6.36.

Proposição 6.3

Se $r_1, r_2, r_3 \in \text{Exp}_\Sigma$, então $r_1(r_2 + r_3) \equiv r_1r_2 + r_1r_3$.

Demonstração Assuma que $r_1, r_2, r_3 \in \text{Exp}_\Sigma$, logo tem-se que;

$$\begin{aligned}
 \mathcal{L}(r_1(r_2 + r_3)) &= \mathcal{L}(r_1)\mathcal{L}((r_2 + r_3)) \\
 &= \mathcal{L}(r_1)(\mathcal{L}(r_2 + r_3)) \\
 &= \mathcal{L}(r_1)(\mathcal{L}(r_2) \cup \mathcal{L}(r_3)) \\
 &= \mathcal{L}(r_1)(\mathcal{L}(r_2) \cup \mathcal{L}(r_3)) \\
 &= \mathcal{L}(r_1)\mathcal{L}(r_2) \cup \mathcal{L}(r_1)\mathcal{L}(r_3) \\
 &= \mathcal{L}(r_1r_2 + r_1r_3)
 \end{aligned}$$

O que conclui a prova. ■

Agora será mostrado qual a classe (ou tipo) de linguagens definidas por expressões regulares, ou seja, agora será mostrado a que classe pertencem as linguagens fruto da valoração das expressões regulares.

Teorema 6.12 (Transformação de expressões regulares para AFN)

Se $L = \mathcal{L}(r)$ para alguma expressão regular r , então existe um λ -AFN A tal que $L = \mathcal{L}(A)$. ♥

Demonstração Suponha que $L = \mathcal{L}(r)$ para alguma expressão regular r , agora por indução sobre o número de operadores em r será mostrado que existe um λ -AFN A tal que $L = \mathcal{L}(A)$.

• Base da indução:

Para as expressões regulares r com 0 operadores, isto é, $r = \lambda$ ou $r = \emptyset$ ou $r = a$ para $a \in \Sigma$ considere os seguintes λ -AFN.

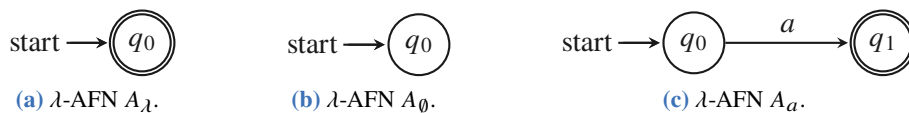


Figura 6.17: Os três λ -AFN básicos para as expressões regulares primitivas.

Claramente tem-se que $\mathcal{L}(\lambda) = \mathcal{L}(A_\lambda)$, $\mathcal{L}(\emptyset) = \mathcal{L}(A_\emptyset)$ e $\mathcal{L}(a) = \mathcal{L}(A_a)$ para todo $a \in \Sigma$.

• **Hipótese indutiva (HI):**

Suponha que para toda $r \in Exp_\Sigma$ com n operadores tal que $n \geq 0$ existe um λ -AFN $A = \langle Q_r, \Sigma, \delta_N^r, q_0^r, \{q_f^r\} \rangle$ tal que $\mathcal{L}(r) = \mathcal{L}(A^r)$. Para fins de representação considere que tal λ -AFN A^r tem a forma a seguir.

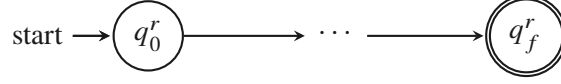


Figura 6.18: λ -AFN A^r para uma expressão r genérica com n operadores.

• **Passo indutivo:**

Agora dado uma expressão regular r com $n + 1$ operadores tal que $n \geq 0$, existe três (e apenas três) casos possíveis para a forma de r .

1º Caso: $r = r_1 + r_2$, assim respectivamente r_1 e r_2 possuem k_1 e k_2 operadores tais que $k_1 + k_2 = n$, obviamente $k_1, k_2 \geq 0$ e, portanto, por (HI) tem-se que existem $A^{r_1} = \langle Q_{r_1}, \Sigma, \delta_N^{r_1}, q_0^{r_1}, \{q_f^{r_1}\} \rangle$ e $A^{r_2} = \langle Q_{r_2}, \Sigma, \delta_N^{r_2}, q_0^{r_2}, \{q_f^{r_2}\} \rangle$ tal que $\mathcal{L}(r_1) = \mathcal{L}(A^{r_1})$ e $\mathcal{L}(r_2) = \mathcal{L}(A^{r_2})$, agora pode-se criar um novo λ -AFN $A^r = \langle Q_{r_1} \cup Q_{r_2} \cup \{q_0\}, \Sigma, \delta_N^r, q_0, \{q_f^{r_1}, q_f^{r_2}\} \rangle$ onde,

$$\delta_N^r(q, a) = \begin{cases} \delta_N^{r_1}(q, a), & \text{se } q \in Q_{r_1} \\ \delta_N^{r_2}(q, a), & \text{se } q \in Q_{r_2} \\ \{q_0^{r_1}, q_0^{r_2}\}, & \text{se } q = q_0, a = \lambda \\ \emptyset, & \text{qualquer outro caso} \end{cases}$$

Para fins de representação considere que tal λ -AFN A^r tem a forma a seguir.

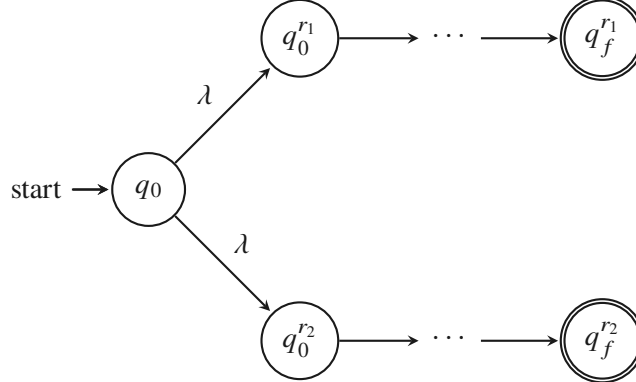


Figura 6.19: λ -AFN A^r para uma expressão $r_1 + r_2$.

Agora note que para todo $w \in \Sigma^*$ tem-se que,

$$\begin{aligned} w \in \mathcal{L}(r) &\iff w \in \mathcal{L}(r_1 + r_2) \\ &\iff w \in \mathcal{L}(r_1) \cup \mathcal{L}(r_2) \\ &\iff w \in \mathcal{L}(r_1) \text{ ou } w \in \mathcal{L}(r_2) \\ &\iff \widehat{\delta_N^{r_1}}(q_0^{r_1}, w) \cap \{q_f^{r_1}\} \neq \emptyset \text{ ou } \widehat{\delta_N^{r_2}}(q_0^{r_2}, w) \cap \{q_f^{r_2}\} \neq \emptyset \\ &\iff \widehat{\delta_N}(q_0, w) \cap \{q_f^{r_1}\} \neq \emptyset \text{ ou } \widehat{\delta_N}(q_0, w) \cap \{q_f^{r_2}\} \neq \emptyset \\ &\iff \widehat{\delta_N}(q_0, w) \cap \{q_f^{r_1}, q_f^{r_2}\} \neq \emptyset \\ &\iff w \in \mathcal{L}(A^r) \end{aligned}$$

2º Caso: $r = r_1 r_2$, assim novamente r_1 e r_2 possuem k_1 e k_2 operadores tais que $k_1 + k_2 = n$,

obviamente $k_1, k_2 \geq 0$ e, portanto, por **(HI)** tem-se que existem $A^{r_1} = \langle Q_{r_1}, \Sigma, \underline{\delta}_N^{r_1}, q_0^{r_1}, \{q_f^{r_1}\} \rangle$ e $A^{r_2} = \langle Q_{r_2}, \Sigma, \underline{\delta}_N^{r_2}, q_0^{r_2}, \{q_f^{r_2}\} \rangle$ tal que $\mathcal{L}(r_1) = \mathcal{L}(A^{r_1})$ e $\mathcal{L}(r_2) = \mathcal{L}(A^{r_2})$, agora pode-se criar um novo λ -AFN $A^r = \langle Q_{r_1} \cup Q_{r_2}, \Sigma, \underline{\delta}_N^r, q_0^r, \{q_f^r\} \rangle$ onde,

$$\underline{\delta}_N^r(q, a) = \begin{cases} \underline{\delta}_N^{r_1}(q, a), & \text{se } q \in Q_{r_1} - \{q_f^{r_1}\}, a \in \Sigma \cup \{\lambda\} \\ \underline{\delta}_N^{r_1}(q, a), & \text{se } q = q_f^{r_1}, a \in \Sigma \\ \underline{\delta}_N^{r_1}(q, a) \cup \{q_0^{r_1}\}, & \text{se } q = q_f^{r_1}, a = \lambda \\ \underline{\delta}_N^{r_2}(q, a), & \text{se } q \in Q_{r_2} - \{q_f^{r_2}\}, a \in \Sigma \cup \{\lambda\} \\ \underline{\delta}_N^{r_2}(q, a), & \text{se } q = q_f^{r_2}, a \in \Sigma \\ \underline{\delta}_N^{r_2}(q, a) \cup \{q_0^{r_2}\}, & \text{se } q = q_f^{r_2}, a = \lambda \\ \emptyset, & \text{qualquer outro caso} \end{cases}$$

E para fins de representação tal λ -AFN A^r possui a forma a seguir.

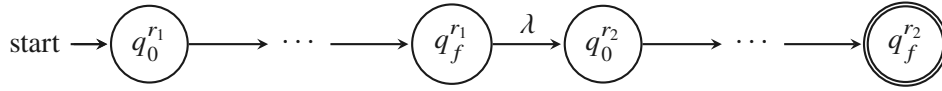


Figura 6.20: λ -AFN A^r para uma expressão $r_1 r_2$.

Agora note que para todo $w \in \Sigma^*$ tal que $w = xy$ com $x, y \in \Sigma^*$ tem-se que,

$$\begin{aligned} w \in \mathcal{L}(r) &\iff w \in \mathcal{L}(r_1 r_2) \\ &\iff xy \in \mathcal{L}(r_1 r_2) \\ &\iff xy \in \mathcal{L}(r_1) \mathcal{L}(r_2) \\ &\iff x \in \mathcal{L}(r_1) \text{ e } y \in \mathcal{L}(r_2) \\ &\iff \widehat{\underline{\delta}_N^{r_1}}(q_0^{r_1}, x) \cap \{q_f^{r_1}\} \neq \emptyset \text{ e } \widehat{\underline{\delta}_N^{r_2}}(q_0^{r_2}, y) \cap \{q_f^{r_2}\} \neq \emptyset \\ &\iff \widehat{\underline{\delta}_N}(q_0^{r_1}, x) \cap \{q_f^{r_1}\} \neq \emptyset \text{ e } \widehat{\underline{\delta}_N}(q_0^{r_2}, y) \cap \{q_f^{r_2}\} \neq \emptyset \\ &\iff \widehat{\underline{\delta}_N}(\widehat{\underline{\delta}_N}(q_0^{r_1}, x), y) \cap \{q_f^{r_2}\} \neq \emptyset \\ &\iff \widehat{\underline{\delta}_N}(q_0^{r_1}, xy) \cap \{q_f^{r_2}\} \neq \emptyset \\ &\iff w \in \mathcal{L}(A^r) \end{aligned}$$

3º Caso: $r = r_1^*$, onde r_1 tem exatamente n operadores sendo que $n \geq 0$, assim por **(HI)** tem-se que existem $A^{r_1} = \langle Q_{r_1}, \Sigma, \underline{\delta}_N^{r_1}, q_0^{r_1}, \{q_f^{r_1}\} \rangle$ tal que $\mathcal{L}(r_1) = \mathcal{L}(A^{r_1})$, agora pode-se criar um novo λ -AFN $A^r = \langle Q_{r_1} \cup \{q_0, q_f\}, \Sigma, \underline{\delta}_N^r, q_0^r, \{q_f\} \rangle$ onde,

$$\underline{\delta}_N^r(q, a) = \begin{cases} \underline{\delta}_N^{r_1}(q, a), & \text{se } q \in Q_{r_1} - \{q_f^{r_1}\} \\ \underline{\delta}_N^{r_1}(q, a), & \text{se } q = q_f^{r_1}, a \in \Sigma \\ \underline{\delta}_N^{r_1}(q, a) \cup \{q_f\}, & \text{se } q = q_f^{r_1}, a = \lambda \\ \{q_0^{r_1}, q_f\}, & \text{se } q = q_0, a = \lambda \\ \{q_0^{r_1}\}, & \text{se } q = q_f, a = \lambda \\ \emptyset, & \text{qualquer outro caso} \end{cases}$$

E para fins de representação tal λ -AFN A^r tem sua forma como descrita pela figura a seguir.

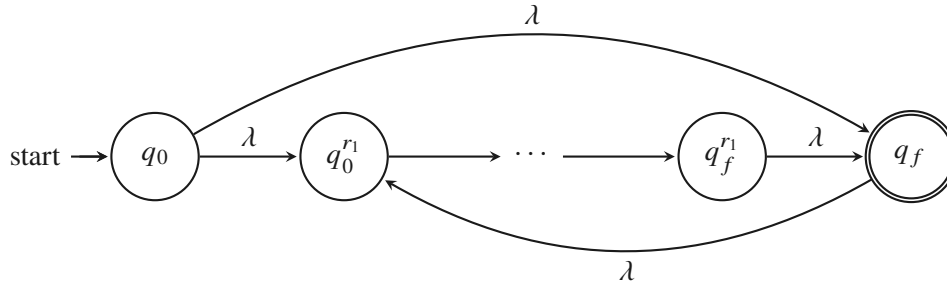


Figura 6.21: λ -AFN A^r para uma expressão r_1^* .

A prova de que $\mathcal{L}(r) = \mathcal{L}(A^r)$ ficará de exercício ao leitor. Agora os três casos anteriores permitem afirmar que sempre existe um λ -AFN A tal que $L = \mathcal{L}(A)$. ■

Uma consequência imediata deste teorema é apresentada a seguir.

Corolário 6.3

A linguagem (ou valoração) de qualquer $r \in \text{Exp}_\Sigma$ é uma linguagem regular. ♡

Demonstração Direto do Teorema 6.12 e o Corolário 6.2. ■

Exemplo 6.39 Dado o alfabeto $\{a, b\}$ e $(ab)^* \in \text{Exp}_\Sigma$ pelo Teorema 6.12 são construídos os autômatos a seguir,

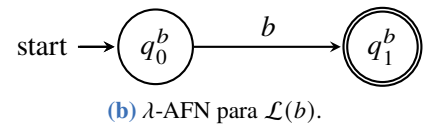
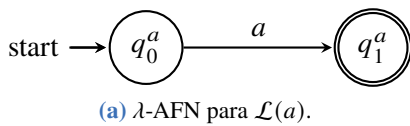


Figura 6.22: Os λ -AFN básicos para as expressões regulares primitivas a e b .

Pode-se então gerar o λ -AFN para reconhecer a concatenação ab , tal autômato será gerado pela combinação dos autômatos das Figuras 6.22a e 6.22b, gerando com resultado o autômato a seguir como se segue,

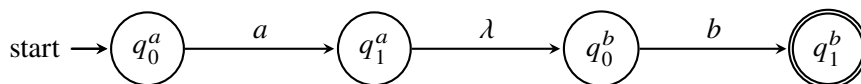


Figura 6.23: λ -AFN para as expressões regulares $\mathcal{L}(ab)$.

Para finalizar é necessário agora apresentar o autômato que reconheça o fecho de Kleene da expressão ab , tal autômato é construído seguindo o Teorema 6.12, e tal autômato é representado pelo grafo de transição esboçado na Figura 6.24 a seguir.

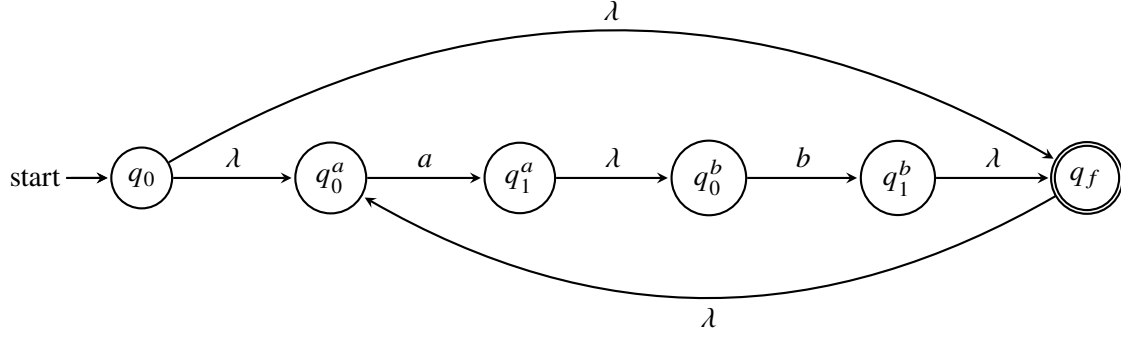


Figura 6.24: λ -AFN para as expressões regular $\mathcal{L}((ab)^*)$.

Observação: Apenas na demonstração a seguir considere que o símbolo Σ representa o somatório enumerável, com respeito a soma (+) das expressões regulares, além disso, para fins de representação durante tal prova a letra X irá representar um alfabeto genérico qualquer.

Teorema 6.13

Se L é uma linguagem regular, então existe uma expressão regular r tal que $L = \mathcal{L}(r)$.

Demonstração Sem perda de generalidade suponha que $L = \mathcal{L}(A)$ para algum λ -AFN $A = \langle \{q_1, \dots, q_n\}, X, \delta, q_1, F \rangle$ em que $n \geq 1$. Agora será construído para todo $k \leq n$ a expressão regular $r_{i,j}^k$ é definida recursivamente como sendo:

$$r_{i,j}^0 = \begin{cases} \sum a, & \text{se } i \neq j \\ \lambda + \sum_{\delta(q_i, a, q_j)} a, & \text{se } i = j \end{cases}$$

$$r_{i,j}^k = (r_{i,k}^{k-1} (r_{k,k}^{k-1})^* r_{k,j}^{k-1}) + r_{i,j}^{k-1}$$

note que $r_{i,j}^k$ é a expressão regular cuja valoração é o conjunto de todas as palavras $w \in X^*$ tal que $\widehat{\delta}(q_i, w) = q_j$ de forma que nenhum estado intermediário q_p com $p > k$ seja usado na computação (para detalhes deste fato consulte [12, 33]). Agora uma vez que A pode ter vários estados $q_f \in F$ tem-se então que cada $w \in X^*$ tal que $\widehat{\delta}(q_1, w) = q_f$ é uma palavra aceita por A , ou seja, $w \in \mathcal{L}(A)$ e, portanto, para $n = \#Q$ a valoração da expressão regular r definida por,

$$r = \sum_{q_f \in F} r_{1,f}^n$$

é exatamente o conjunto de todas as palavras aceitas por A , ou seja, $\mathcal{L}(A) = \mathcal{L}(r)$. ■

Exemplo 6.40 Considere o AFD esboçado pela Figura 6.25,

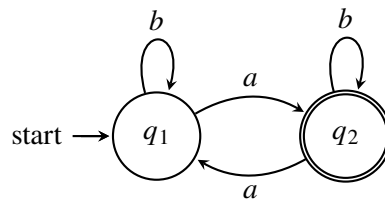


Figura 6.25: Um AFD que computa sobre o alfabeto $\{a, b\}$.

Como $n = \#Q$ tem-se que $n = 2$, e portanto, a expressão regular cuja valoração corresponde a linguagem

do AFD será dada por,

$$r_{1,2}^2 = (r_{1,2}^1 (r_{2,2}^1)^* r_{2,2}^1) + r_{1,2}^1 \quad (6.39)$$

Note porém que,

$$\begin{aligned} r_{1,2}^1 &= (r_{1,1}^0 (r_{1,1}^0)^* r_{1,2}^0) + r_{1,2}^0 \\ &\equiv ((\lambda + b)(\lambda + b)^* a) + a \\ &\equiv b^* a \end{aligned} \quad (6.40)$$

e

$$\begin{aligned} r_{2,2}^1 &= (r_{2,1}^0 (r_{1,1}^0)^* r_{1,2}^0) + r_{2,2}^0 \\ &\equiv (a(\lambda + b)^* a) + (\lambda + b) \\ &\equiv ab^* a + \lambda + b \end{aligned} \quad (6.41)$$

substituindo as Equações (6.40) e (6.41) na Equação (6.39) tem-se que,

$$\begin{aligned} r_{1,2}^2 &= (b^* a (ab^* a + \lambda + b)^* ab^* a + \lambda + b) + (b^* a) \\ &\equiv b^* a (ab^* a + b)^* (ab^* a + \lambda) + b^* a \\ &\equiv b^* a (ab^* a + b)^* + b^* a \\ &\equiv b^* a (ab^* a + b)^* \end{aligned}$$

6.8 Gramática Regulares

Nas seções anteriores foram apresentados dois formalismos para as linguagens regulares, a saber, formalismo operacional (os autômatos) e o formalismo denotacional (as expressões regulares). Nesta seção será apresentado um terceiro formalismo para as linguagens regulares, sendo este um formalismo gerador (ou axiomático) [45].

Definição 6.30 (Gramática Linear)

Uma gramática formal $G = \langle V, \Sigma, S, P \rangle$ é dita *Linear à Direita*, ou simplesmente *GLD*, se todas as suas produções são da forma,

$$A \triangleright wB$$

e é dita *Linear à Esquerda*, ou simplesmente *GLE*, se todas as suas produções são da forma,

$$A \triangleright Bw$$

onde $A \in V, B \in V \cup \{\lambda\}$ e $w \in \Sigma^*$.



Exemplo 6.41 A gramática $G_1 = \langle \{B, S, A\}, \{a, b\}, S, P \rangle$ onde P é formado pelas regras:

$$S \triangleright aaB$$

$$B \triangleright bb$$

é uma GLD.

Exemplo 6.42 A gramática $G_1 = \langle \{X, S, Y\}, \{0, 1\}, S, P \rangle$ onde P é formado pelas regras:

$$S \triangleright X001$$

$$S \triangleright Y011$$

$$X \triangleright S01$$

$$Y \triangleright \lambda$$

é uma GLE.

Em uma gramática formal G quando para uma palavra w existem w_1, \dots, w_n tal que há as seguintes regras $w \triangleright w_1, \dots, w \triangleright w_n \in P$, é comum para simplificar a escrita do conjunto de regras usar a notação $w \triangleright w_1 \mid \dots \mid w_n$.

Exemplo 6.43 Considere a GLE apresentada no Exemplo 6.42 o conjunto P da mesma poderia ser escrito como:

$$S \triangleright X001 \mid Y011$$

$$X \triangleright S01$$

$$Y \triangleright \lambda$$

Definição 6.31 (Gramática Regular)

Uma gramática formal G é dita regular sempre que ela for linear à esquerda ou à direita.

Um tipo mais rigoroso de gramática regular como comentado em [12, 35], são as chamadas gramática regulares unitárias definidas a seguir.

Definição 6.32 (Gramáticas Regulares Unitárias)

Uma gramática regular G é dita unitária à esquerda (à direita) se ela é linear à esquerda (à direita) e toda produção é da forma $A \triangleright Bw$ ($A \triangleright wB$) com $A \in V, B \in V \cup \{\lambda\}$ e $w \in \Sigma \cup \{\lambda\}$.

Exemplo 6.44 A gramática G do Exemplo 6.43 é uma gramática regular pois é uma GLE, porém não é uma gramática regular unitária.

Observação: De forma natural como o leitor deve estar pensando agora, duas gramática G_1 e G_2 serão ditas equivalentes sempre que elas gerarem a mesma linguagens.

O próximo resultado estabelece que gramática regulares e gramática regulares unitárias tem o mesmo poder de geração de linguagens.

Teorema 6.14

$L = \mathcal{L}(G)$ para alguma gramática regular G se, e somente se, existe uma gramática regular unitária G' na mesma direção (esquerda ou direita) tal que $L = \mathcal{L}(G')$.

Demonstração (\Rightarrow) Suponha que $L = \mathcal{L}(G)$ para alguma gramática regular $G = \langle V, \Sigma, S, P \rangle$ tal que G seja linear à esquerda (a prova é similar para o caso à direita). Agora construa uma nova gramática $G' = \langle V', \Sigma, S, P' \rangle$ tal que P' é definido usando as seguintes regras:

R1: Se $A \triangleright Bw \in P$ onde $B \in V \cup \{\lambda\}, |w| \leq 1$, então $A \triangleright Bw \in P'$.

R2: Se $A \triangleright Ca_1 \dots a_n \in P$ onde $B \in V$ e $a_i \in \Sigma$ sendo $1 \leq i \leq n$ e $n > 1$, então tem-se que $A \triangleright B_n a_n, B_n \triangleright B_{n-1} a_{n-1}, \dots, B_2 \triangleright Ca_1 \in P'$.

R3: Se $A \triangleright a_1 \dots a_n \in P$ onde $a_i \in \Sigma$ sendo $1 \leq i \leq n$ e $n \geq 2$, então tem-se que $A \triangleright B_n a_n, B_n \triangleright$

$$B_{n-1}a_{n-1}, \dots, B_2 \triangleright B_1a_1, B_1 \triangleright \lambda \in P'.$$

Para as regras R2 e R3 todo B_i é uma nova variável existente em V' que não existe originalmente em V . Claramente a gramática G' é regular unitária à esquerda. Também não é difícil mostra por indução sobre o tamanho das derivações que para todo $w \in \Sigma^*$ tem-se que $S \gg_G^* w$ se, e somente se, $S \gg_{G'}^* w$, portanto, $\mathcal{L}(G) = \mathcal{L}(G')$.

(\Leftarrow) Trivial uma vez que toda gramática regular unitária à esquerda (à direita) é um caso particular de gramática regular à esquerda (à direita). ■

Exemplo 6.45 Considerando a gramática regular do Exemplo 6.43 usando o Teorema 6.14 é gerado a gramática regular unitária $G' = \langle \{S, B_3, B_2, C_3, C_2, X, D_2, Y\}, \{0, 1\}, S, P' \rangle$ onde P' é formado pelas regras:

$$\begin{aligned} S &\triangleright B_31 \mid C_31 \\ B_3 &\triangleright B_20 \\ B_2 &\triangleright X0 \\ C_3 &\triangleright C_21 \\ C_2 &\triangleright Y0 \\ X &\triangleright D_21 \\ D_2 &\triangleright S0 \\ Y &\triangleright \lambda \end{aligned}$$

Observação: Obviamente a gramática do Exemplo 6.45 poderia ser otimizada para usar menos variáveis, porém otimização não é o foco de interesse no Teorema 6.14.

O próximo resultado estabelece o poder de geração das gramáticas regulares à direita.

Teorema 6.15

$L = \mathcal{L}(G)$ para alguma gramática regular à direita G se, e somente se, L é uma linguagem regular.

Demonstração (\Rightarrow) Suponha que $L = \mathcal{L}(G)$ para alguma gramática regular à direita G , assim pelo Teorema 6.14 existe uma gramática regular unitária à direita $G' = \langle V, \Sigma, S, P \rangle$ tal que $L = \mathcal{L}(G')$, sem perda de generalidade¹¹ pode-se assumir que toda regra em P é da forma $A \triangleright aB$ ou $A \triangleright \lambda$ com $A, B \in V$ e $a \in \Sigma \cup \{\lambda\}$, dito isto, pode-se agora construir um λ -AFN $M = \langle V \cup \{q_f\}, \Sigma, \delta_N, S, \{q_f\} \rangle$ tal que:

$$\begin{aligned} B \in \delta_N(A, a) &\iff A \triangleright aB \in P \\ q_f \in \delta_N(A, \lambda) &\iff A \triangleright \lambda \in P \end{aligned}$$

Agora será mostrado por indução sobre o tamanho das derivações em G' que se w é derivada por G' e $w \in \Sigma^*$, então é aceita por M .

• Base da indução:

Quando w é derivada em G' com uma única derivação tem-se então duas situações possíveis:

- (1) Quando $w = \lambda$, obrigatoriamente existe uma regra da forma $S \triangleright \lambda$, e pela construção de M tem-se que $q_f \in \delta_N(S, \lambda)$, logo $\widehat{\delta_N}(S, \lambda) \cap \{q_f\} \neq \emptyset$ e, portanto, $\lambda \in L(M)$.
- (2) Quando $w = aB$, existe em P uma regra da forma $S \triangleright aB$ com $a \in \Sigma \cup \{\lambda\}$ e $B \in V$, assim pela construção de M tem-se que $B \in \delta_N(S, a)$. Como $aB \notin \Sigma^*$ não há mais nada a fazer nesse caso.

• Hipótese indutiva (HI):

¹¹Basta gerar uma nova gramática onde toda regra da forma $A \triangleright a$ com $a \in \Sigma$ foi substituída pelas regras $A \triangleright aC$ e $C \triangleright \lambda$ onde C é uma variável nova criada, obviamente a nova gramática continua equivalentes a antiga.

Suponha que $S \gg_{G'}^* w$ em n derivação com $n \geq 1$ tal que:

- (1) Se $w \in \Sigma^*$, então $w \in \mathcal{L}(M)$.
- (2) Se $w = a_1 \cdots a_{n-1}B$ com $a_i \in \Sigma \cup \{\lambda\}$ para todo $1 \leq i \leq n-1$ e $B \in V$, então $B \in \widehat{\delta_N}(S, a_1 \cdots a_{n-1})$.

• **Passo indutivo:**

Agora dado que $S \gg_{G'}^* w'$ em $n+1$ derivações, tem-se obrigatoriamente que acontece o caso (2) de **(HI)** e nesse caso duas situações são possíveis:

- (1) Se $w' \in \Sigma^*$, então $w = a_1 \cdots a_{n-1}B$ com $a_i \in \Sigma \cup \{\lambda\}$ para todo $1 \leq i \leq n-1$ e existe em P uma produção $B \rightarrow \lambda$, e assim, $w' = a_1 \cdots a_{n-1}$, nesta situação pelo caso (2) de **(HI)** tem-se que $B \in \widehat{\delta_N}(S, a_1 \cdots a_{n-1})$ e como $B \rightarrow \lambda$ pela construção de M tem-se que $q_f \in \delta_N(B, \lambda)$, consequentemente, $\widehat{\delta_N}(S, a_1 \cdots a_{n-1}) \cap \{q_f\} \neq \emptyset$ e, portanto, $w' \in \mathcal{L}(M)$.
- (2) Se $w' = a_1 \cdots a_{n-1}B$ com $a_i \in \Sigma \cup \{\lambda\}$ para todo $1 \leq i \leq n-1$ e $B \in V$, então pela construção de M tem-se que $B \in \widehat{\delta_N}(S, w)$ como $w' \notin \Sigma^*$ não há mais nada a fazer nesse caso.

Portanto, o raciocínio indutivo anterior garante que sempre que w é derivada por G' e $w \in \Sigma^*$ tem-se que $w \in \mathcal{L}(M)$ e assim pode-se afirmar pelo Corolário 6.2 que L é regular. (\Leftarrow) Suponha que L é uma linguagem regular assim por definição existe um AFD $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ tal que $L = \mathcal{L}(M)$, assim construa uma gramática regular unitária à direita $G = \langle Q, \Sigma, q_0, P \rangle$ onde o conjunto P é definido usando as regras a seguir,

- (a) Se $\delta(q_i, a) = q_j$, então $q_i \triangleright aq_j \in P$.
- (b) Se $q_i \in F$, então $q_i \triangleright \lambda \in P$.

Agora note que para todo $w \in \Sigma^*$ com $w = a_1 \cdots a_n$ tem-se que,

$$\begin{aligned}
 w \in \mathcal{L}(M) &\iff \widehat{\delta}(q_0, w) \in F \\
 &\iff \widehat{\delta}(q_0, a_1 \cdots a_n) \in F \\
 &\iff (\exists q_f \in F) [\widehat{\delta}(q_0, w) = q_f] \\
 &\iff (\exists q_1, \dots, q_{n-1} \in Q, q_f \in F) [\delta(q_0, a_1) = q_1 \wedge \cdots \wedge \delta(q_{n-1}, a_n) = q_f] \\
 &\iff (\exists q_1, \dots, q_{n-1} \in Q, q_f \in F) [q_0 \triangleright a_1 q_1, \dots, q_{n-1} \triangleright a_n q_f, q_f \triangleright \lambda \in P] \\
 &\iff q_0 \gg^* a_1 \cdots a_n \\
 &\iff q_0 \gg^* w \\
 &\iff w \in \mathcal{L}(G)
 \end{aligned}$$

portanto, $\mathcal{L}(M) = \mathcal{L}(G)$ o que conclui a prova. ■

Lema 6.5

Se L é gerada por uma gramática à direita, então L^r é gerada por uma gramática regular à direita. ♥

Demonstração Suponha que L é gerada por uma gramática à direita G , ou seja, $L = \mathcal{L}(G)$, assim pelo Teorema 6.15 tem-se que L é regular, logo existe um AFD $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ tal que $L = \mathcal{L}(M)$, agora construa um λ -AFN $M_1 = \langle Q \cup \{q_f\}, \Sigma, \delta_N, q_0, \{q_f\} \rangle$ tal que,

$$\delta_N(q, a) = \begin{cases} \{\delta(q, a)\}, & \text{se } q \in Q, a \in \Sigma \\ \{q_f\}, & \text{se } q \in F, a = \lambda \\ \emptyset, & \text{qualquer outro caso} \end{cases}$$

claramente $L = \mathcal{L}(M_1)$, agora construa um novo λ -AFN $M_2 = \langle Q \cup \{q_f\}, \Sigma, \delta'_N, q_f, \{q_0\} \rangle$ onde para todo $q \in Q \cup \{q_f\}$ e $a \in \Sigma \cup \{\lambda\}$ tem-se que,

$$q_i \in \delta'_N(q_j, a) \iff q_j \in \delta_N(q_i, a)$$

pela construção de M_2 é claro que $w \in \mathcal{L}(M_1) \iff w^r \in \mathcal{L}(M_2)$ e, portanto, $L^r = \mathcal{L}(M_2)$. Desde que M_2 é um λ -AFN pelo Corolário 6.2 tem-se que L^r é uma linguagem regular, consequentemente, pelo Teorema 6.15 existe uma gramática regular à direita G' tal que $L = \mathcal{L}(G')$, o que conclui a prova. ■

O próximo resultado mostra que gramáticas regulares à esquerda e à direita são equivalentes.

Teorema 6.16 (Mudança de direção regular)

L é gerada por uma gramática à esquerda se, e somente se, L é gerada por uma gramática regular à direita.

Demonstração (\Rightarrow) Suponha que L é gerada por uma gramática à esquerda $G = \langle V, \Sigma, S, P \rangle$, pelo Teorema 6.14 pode-se assumir que G é uma gramática regular unitária também à esquerda, assim todas as regras em P são da forma $A \triangleright Ba$ com $A \in V, B \in V \cup \{\lambda\}$ e $a \in \Sigma \cup \{\lambda\}$. Sem perda de generalidade¹² pode-se construir uma nova gramática regular unitária à esquerda $G' = \langle V', \Sigma, S, P' \rangle$ onde toda regra em P' é da forma $A \triangleright Ba$ ou $A \triangleright \lambda$ com $A, B \in V'$ e $a \in \Sigma \cup \{\lambda\}$ claramente pela construção de G' tem-se que $\mathcal{L}(G) = \mathcal{L}(G')$, agora construa um λ -AFN $M = \langle V' \cup \{q_f\}, \Sigma, \delta_N, S, \{q_f\} \rangle$ onde,

$$B \in \delta_N(A, a) \iff A \triangleright Ba \in P$$

$$q_f \in \delta_N(A, \lambda) \iff A \triangleright \lambda \in P$$

Agora note que para todo $w = a_1 \cdots a_n \in \Sigma^*$ tem-se que,

$$\begin{aligned} w \in \mathcal{L}(G') &\iff a_1 \cdots a_n \in \mathcal{L}(G') \\ &\iff S \gg_{G'}^* a_1 \cdots a_n \\ &\iff (\exists A_1 \cdots A_n, S \in V) \\ &\quad [S \triangleright A_n a_n, A_n \triangleright A_{n-1} a_{n-1}, \dots, A_2 \triangleright A_1 a_1, A_1 \triangleright \lambda \in P'] \\ &\iff (\exists A_1 \cdots A_n, S \in V) \\ &\quad [A_n \in \delta_N(S, a_n), A_{n-1} \in \delta_N(A_n, a_{n-1}), \dots, A_1 \in \delta_N(A_2, a_1), \\ &\quad q_f \in \delta_N(A_1, \lambda)] \\ &\iff a_n \cdots a_1 \in \mathcal{L}(M) \\ &\iff w^r \in \mathcal{L}(M) \end{aligned}$$

Logo $\mathcal{L}(M) = \mathcal{L}(G')^r$, desde que M é um λ -AFN tem-se pelo Corolário 6.2 que $\mathcal{L}(G')^r$ é regular, assim pelo Teorema 6.15 existe uma gramática regular à direita \hat{G}_1 que a gera, ou seja, $\mathcal{L}(\hat{G}_1) = \mathcal{L}(G')^r$, mas pelo Lema 6.5 irá existir outra gramática regular à direita \hat{G}_2 tal que $\mathcal{L}(\hat{G}_2) = \mathcal{L}(\hat{G}_1)^r$, mas $\mathcal{L}(\hat{G}_1)^r = (\mathcal{L}(G')^r)^r = (\mathcal{L}(G)^r)^r = (L^r)^r = L$, portanto, L é gerada por uma gramática regular à direita. (\Leftarrow) Suponha que L é gerada por uma gramática regular à direita, ou seja, que existe uma gramática regular à direita G tal que $L = \mathcal{L}(G)$, assim pelo Lema 6.5 irá existir outra gramática regular à direita $G_1 = \langle V, \Sigma, S, P \rangle$ tal que $L^r = \mathcal{L}(G_1)$, sem perda de generalidade pelo Teorema 6.14 pode-se assumir que G_1 é regular unitária à direita, logo todas as suas produções são da forma $A \triangleright aB$ com $A \in V, B \in V \cup \{\lambda\}$ e $a \in \Sigma \cup \{\lambda\}$. Dito isso construa uma nova gramática $G_2 = \langle V, \Sigma, S, P' \rangle$ onde $P' = \{A \triangleright Ba \mid A \triangleright Ba \in P\}$, claramente G_2 é unitária à esquerda. Mas pela construção de G_2 fica claro que $S \gg_{G_1}^* w \iff S \gg_{G_2}^* w^r$, logo $\mathcal{L}(G_1)^r = \mathcal{L}(G_2)$, mas desde que, $\mathcal{L}(G_1)^r = (L^r)^r = L$, tem-se então que L é gerada por uma gramática linear à esquerda, o que completa a

¹²Basta gerar uma nova gramática onde toda regra da forma $A \triangleright a$ com $a \in \Sigma$ foi substituída pelas regras $A \triangleright Ca$ e $C \triangleright \lambda$ onde C é uma variável nova criada.

prova. ■

Exemplo 6.46 Dado a gramática regular à direita $G_1 = \langle \{A, B, C\}, \{a, b\}, A, P_1 \rangle$ com P_1 é formado pelas seguintes regras,

$$A \triangleright aC \mid B$$

$$B \triangleright bB \mid \lambda$$

$$C \triangleright aA$$

claramente $\mathcal{L}(G_1) = \{w \in \{a, b\}^* \mid w = a^{2m}b^n \text{ com } m, n \in \mathbb{N}\}$, agora usando a construção exposta pelo Teorema 6.16, é possível construir a gramática regular à esquerda $G_2 = \langle \{A, B, C\}, \{a, b\}, B, P_2 \rangle$ onde P_2 é formado pelas seguintes regras,

$$B \triangleright Bb \mid Ab \mid A$$

$$A \triangleright Ca \mid \lambda$$

$$C \triangleright Aa$$

e obviamente $\mathcal{L}(G_2) = \{w \in \{a, b\}^* \mid w = a^{2m}b^n \text{ com } m, n \in \mathbb{N}\}$.

Observação: Para esse questionário sempre que $w \in \Sigma^*$ e $c \in \Sigma$ a notação $|w|_c$ irá representar o número de c 's que existem na palavra w .

Questionário do Capítulo

1. Considere o autômato na Figura 6.26 e responda o que é solicitado.

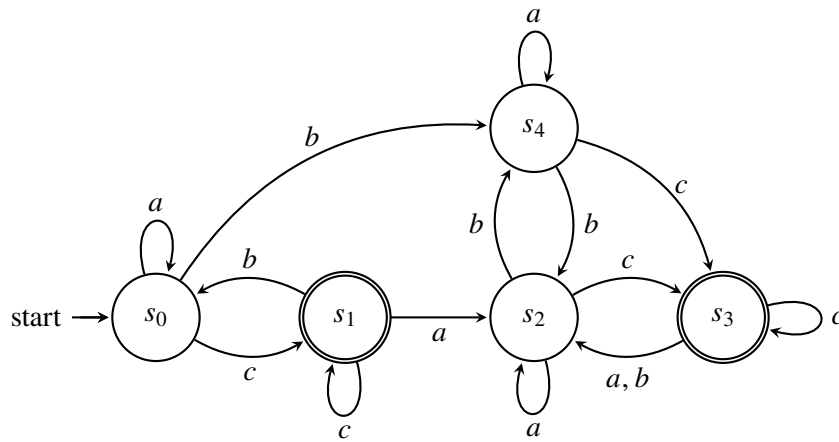


Figura 6.26: Autômato para o exercício 1.

- (a). Verifique se as palavras: $abccaaabacab$, $ccccbacabacbb$, $bbacabb$, $aaccca$, $aaabb$, $acacabb$, $bbacac$ e $ccabbbacac$ são aceitas ou não pelo autômato.
 - (b). λ é aceito por este autômato?
 - (c). Traduza o grafo de transição para a notação algébrica.
2. Construa um AFD que compute cada linguagem a seguir.
 - (a). $L_1 = \{\lambda, 00101111, 11001101, 1\}$.
 - (b). $L_2 = \{w10 \in \{0, 1\}^* \mid w = (001)^n 11 \text{ com } n \in \mathbb{N}\}$.
 - (c). $L_3 = \{w \in \{a, b\}^* \mid w = a^{2m}b^{2n+1} \text{ ou } w = aab^{3m+3}b^n \text{ com } m, n \in \mathbb{N}\}$.
 - (d). $L_4 = \{w \in \{a, b, c, d\}^* \mid w \text{ não contém as subcadeias } ab \text{ e } cd\}$.
 - (e). $L_5 = \{w \in \{0, 1\}^* \mid (\forall n \in \mathbb{N})[|w|_1 \neq 3n]\}$.

- (f). $L_6 = \{w \in \{0, 1\}^* \mid w = x_1 \cdots x_n \text{ e } x_i = 0 \text{ se } i \text{ for par, senão } x_i = 1, \text{ sendo } n \in \mathbb{N}\}$.
- (g). $L_7 = \{w \in \{x, y, z\}^* \mid \text{Se } w \text{ contém a sub-palavra } zz, \text{ então à direita da sub-palavra } zz \text{ não ocorre a sub-palavra } zz\}$.
- (h). $L_8 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\} \cup \{bbca, ccab, ccab, baba\}$.
- (i). $L_9 = \{uv \in \{0, 1\}^* \mid u = 1^m 0111, v = 0100^p 1 \text{ com } m, p \in \mathbb{N}\}$.
- (j). $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ é um número binário múltiplo de } 3\}$.
3. Considerando o alfabeto $\Sigma = \{c, d\}$ para cada uma das linguagens definidas pelas propriedades a seguir construa um AFD que a compute.
- w possui exatamente um único símbolo d , e este não aparece no final das palavras.
 - w tem apenas uma única sub-palavra dd .
 - w não contém três d 's seguidos.
 - w possui pelo menos um c .
 - w possui 6 ou menos símbolos, e não existe as sub-palavras cc e dd .
 - w tem mais que 4 símbolos c .
 - w é da forma db^4wb^5d com $w \in \Sigma^*$.
 - w possui tamanho n tal que $n \bmod 3 \neq 0$.
 - $|w| = n$ com $n \geq 3$ e $|w|_c \bmod 2 > 1$.
 - w é qualquer palavra tal que $3 \leq |w|_c \leq 6$
 - w tem a forma $c^m d^n$ tal que m ou n não divisível por 2.
 - w tem a forma $c^m d^n c^p$ tal que $mnp > 6$ e mnp seja ímpar.
 - w é tal que $|w|_d = 2$ ou $|w|_c = 1$.
 - w tem a forma $c^m d^n$ tal que $m + n > mn$.
 - w possui a sub-palavra $ddcd$ e $|w|_d + |w|_c > 6$.
4. Considerando o alfabeto $\Sigma = \{2, 3, 5\}$, demonstre usando AFD que as linguagens definidas pelas propriedades a seguir são regulares.
- $L_{pp} = \{w \in \Sigma^* \mid |w|_2, |w|_5 \text{ são ambos pares}\}$.
 - $L_{pi} = \{w \in \Sigma^* \mid |w|_2 \cdot |w|_3 \text{ é par e } |w|_5 \text{ é ímpar}\}$.
 - $L_{pip} = \{w \in \Sigma^* \mid |w|_2 \text{ é par, } |w|_3 \text{ é ímpar e } |w|_5 \text{ é par}\}$.
 - $L_{pu} = \{w \in \Sigma^* \mid w = x_1 \cdots x_n, x_1 = x_n \text{ com } n \in \mathbb{N}\}$.
 - $L_{pud} = \{w \in \Sigma^* \mid w = x_1 \cdots x_n, x_1 \neq x_n \text{ com } n \in \mathbb{N}\}$.
 - $L_{dif} = \{wuv \in \Sigma^* \mid w, v \in \{3, 5\}^*, |u| \geq 3\}$.
 - $L_{inv} = \{wuv \in \Sigma^* \mid w \in \{2\}^*, v \in \{3, 5\}^+, u \in \{3\}^*\}$.
5. Dado um AFD $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ qualquer, mostre que para todo $u, v \in \Sigma^*$ tem-se que $\widehat{\delta}(q_0, uv) = \widehat{\delta}(\widehat{\delta}(q_0, u), v)$.
6. Considerando os AFN das Figuras 6.5, 6.7 e 6.8 execute as computações das palavras $aabbababa$, $bbababba$, $bbbbabaabb$, $bababaaa$ e $ababababb$.
7. Considerando o AFN da Figura 6.9 construa as árvores de computação para as palavras 010101 e 11001101.
8. Converta os AFN das Figuras 6.5, 6.7, 6.8 e 6.9 para a notação algébrica.
9. Encontre os AFD equivalentes aos AFN das Figuras 6.5, 6.7 e 6.8.
10. Seja $L = \{01, 012\}$, construa um AFN com 4 estados (ou menos) que aceite a linguagem L^* .
11. Dado a linguagem $L = \{0101^m \mid m \geq 1\} \cup \{010^n \mid n \in \mathbb{N}\}$ construa um AFN com 5 ou menos estados que aceite a linguagem L .
12. Dado os AFN que você construiu nos Exercícios 10 e 11, encontre os AFD equivalentes a eles.

13. Considerando o AFN representado pelo grafo de transição na Figura 6.27 responda o que é solicitado.

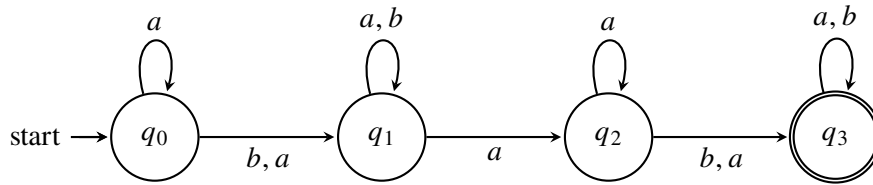


Figura 6.27: Autômato para o exercício 12.

- Realize a computação para as palavras *aaba* e *baba*.
 - Esboce a árvore de computação para a palavra *abbaabaab*.
 - Converta o AFN da Figura 6.27 para a notação algébrica.
 - Encontre um AFD equivalente ao AFN da Figura 6.27.
14. Considerando o AFN representado pelo grafo de transição na Figura 6.28 responda o que é solicitado.

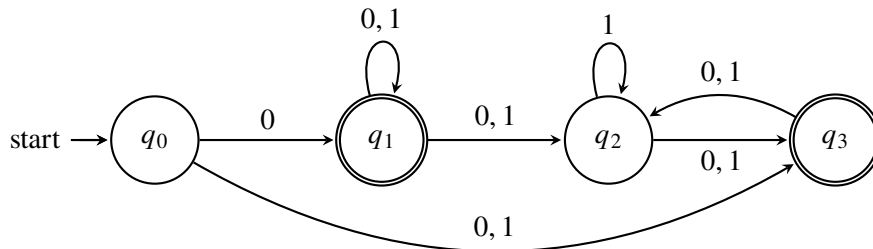


Figura 6.28: Autômato para o exercício 14.

- Realize a computação para as palavras 01110 e 1001011.
 - Esboce a árvore de computação para a palavra 110110101.
 - Converta o AFN da Figura 6.28 para a notação algébrica.
 - Encontre um AFD equivalente ao AFN da Figura 6.28.
15. Considerando o λ -AFN do Exemplo 6.24 responda o que é solicitado.
- Calcule $\delta_\lambda(q_0)$, $\delta_\lambda(q_1)$ e $\delta_\lambda(q_2)$.
 - Compute $\widehat{\delta_N}(q_0, 11233)$.
 - Apresente um AFD equivalente ao λ -AFN.
16. Considerando $\Sigma_1 = \{0, 1\}$ e $\Sigma_2 = \{2, 3\}$ construa um λ -AFN com um único estado final que aceite a linguagem $\{w \in \Sigma_1^* \mid |w|_1 = 2i, i \in \mathbb{N}\} \cup \{w \in \Sigma_2^* \mid |w|_3 = 2j + 1, j \in \mathbb{N}\}$.
17. Considere o alfabeto $\Sigma = \{a, b, c\}$ e as linguagens $L_1 = \{w \in \Sigma^* \mid |w|_a \geq 0, |w|_b > 0, |w|_c = 2k, k \in \mathbb{N}\}$ e $L_2 = \{abaccc, ababc, abacab, acbcc, bacbaaa, abb, bba, aaaabbbb, cac, ccba, caccabcac\}$. Para cada uma das linguagens $L_1 \cup L_2$, $L_1 L_2$ e $L_2 L_1$ construa um λ -AFN A que as aceite.
18. Considerando o λ -AFN $A = \langle Q, \Sigma, \delta_N, q_0, \{q_2\} \rangle$ onde δ_N é especificada pela Tabela 6.7, responda o que é solicitado.

$Q' \backslash \Sigma$	λ	a	b	c
q_0	\emptyset	$\{q_0\}$	$\{q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_1\}$	$\{q_2\}$	\emptyset
q_2	$\{q_1\}$	$\{q_2\}$	\emptyset	$\{q_0\}$

Tabela 6.7: Tabela da função de transição do λ -AFN do Exercício 18.

- (a). Para cada estado do autômato calcule δ_λ .
- (b). Esboce todos os w pertencentes a linguagem do autômato tal que $|w| \leq 4$.
- (c). Encontre um AFD equivalente A .
19. Considerando o λ -AFN $S = \langle Q, \Sigma, \delta_N, s_0, \{s_2\} \rangle$ onde δ_N é especificada pela Tabela 6.8, responda o que é solicitado.

$\Sigma \backslash Q'$	λ	0	1	2
s_0	$\{s_1, s_2\}$	\emptyset	$\{s_1\}$	$\{s_2\}$
s_1	\emptyset	$\{s_0\}$	$\{s_2\}$	$\{s_0, s_2\}$
s_2	\emptyset	\emptyset	\emptyset	\emptyset

Tabela 6.8: Tabela da função de transição do λ -AFN do exercício 19.

- (a). Para cada estado do autômato calcule δ_λ .
- (b). Esboce todos os w pertencentes a linguagem do autômato tal que $|w| \leq 3$.
- (c). Encontre um AFD equivalente S .
20. Para as linguagens especificadas pelos enunciados a seguir construa λ -AFN que as aceite.
- (a). A linguagem de todas as palavras que começam e terminam com qualquer letra do alfabeto $\{a, b, c\}$, porém não existe nas palavras dois a 's e dois c 's seguidos.
- (b). A linguagem de todas as palavras sobre o alfabeto $\{0, 1\}$ que iniciam com o prefixo $(01)^n$ e termina com o sufixo 10 ou começam com o prefixo $(101)^n$ e terminam com o sufixo 00, sendo $n \in \mathbb{N}$ tal que $n \geq 1$.
- (c). A linguagem de todas as palavras sobre o alfabeto $\{x, y\}$ tal que no mínimo uma das quatro últimas letras da palavra é um x .
- (d). A linguagem de todas as palavras sobre o alfabeto $\{x, y, z\}$ tal que no máximo uma das três primeiras letras da palavra é um y ou as duas primeiras são z .
21. Encontre o AFD quociente do AFD apresentado no Exercício 1.
22. Encontre o AFD quociente equivalente ao AFN apresentado no Exercício 13.
23. Encontre o AFD quociente equivalente ao AFN apresentado no Exercício 14.
24. Encontre o AFD quociente equivalente ao λ -AFN apresentado no Exercício 18.
25. Encontre o AFD quociente equivalente ao λ -AFN apresentado no Exercício 19.
26. Encontre o AFD quociente equivalente ao λ -AFN da Figura 6.29 a seguir.

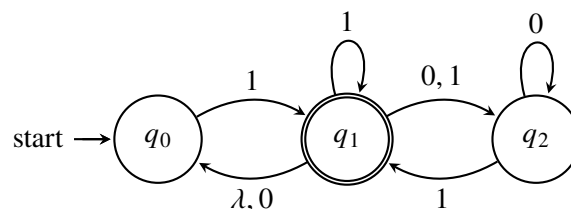


Figura 6.29: Autômato para o Exercício 26.

27. Encontre o AFD quociente equivalente ao λ -AFN da Figura 6.30 a seguir.

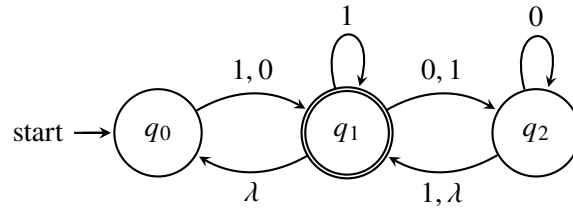


Figura 6.30: Autômato para o Exercício 27.

28. Mostre uma expressão regular cuja valoração é exatamente a linguagem $\{aabb, aaabbb, aaba, \lambda\}$.
29. Construa uma expressão regular para cada uma das linguagens descrita no Exercício 3.
30. Construa uma expressão regular para cada uma das linguagens descrita no Exercício 20.
31. Para cada um dos AFD A , que foram criados por você para responder o Exercício 2, determine uma expressão regular r tal que $\mathcal{L}(r) = \mathcal{L}(A)$.
32. Para cada expressão regular a seguir construa um autômato que aceite a linguagem da expressão regular.
- $a^*b + a$.
 - $(ab + cd)^*$.
 - $(a^*ab)^* + bc$.
 - $(aa)^*(b + \lambda) + \emptyset$.
 - $(a^*) + b(aa)^*$.
33. Dado o alfabeto $\{a, b\}$ demonstre as asserções a seguir.
- $(a + b)^* \equiv (a^*b^*)^*$.
 - $a^* + b^* \not\equiv (a + b)^*$.
 - $a^*b^* \not\equiv (ab)^*$.
 - $(b + ab)^*(a + \lambda) \equiv (a + \lambda)(ba + b)^*$.
 - $a^*a \equiv aa^*$.
 - $\emptyset + (a + b)^* \equiv (a + b)^* + \emptyset$.
34. Para quais quer $r_1, r_2, r_3 \in \text{Exp}_\Sigma$ demonstre as asserções a seguir.
- $(r_1^*)^* \equiv r_1^*$.
 - $(r_1 + r_2) \equiv r_2 + r_1$.
 - $r_1 + (r_2 + r_3) \equiv (r_1 + r_2) + r_3$.
 - $r_1 + r_1 \equiv r_1$.
 - $(r_1 + r_2)^* \equiv (r_1^*r_2^*)^*$.
 - $r_1r_1 \equiv r_1$.
 - $(r_1r_2)^*r_1 \equiv r_1(r_2r_1)^*$.
 - $r_1 + \emptyset \equiv r_1$.
35. Construa um autômato que aceite cada valoração a seguir.
- $\mathcal{L}(aa^*(b + a))$.
 - $\mathcal{L}((aa + abb)^*(aa + \lambda + ba))$.
 - $\mathcal{L}((ab + b)^*(a + \lambda))$.
 - $\mathcal{L}(aa^* + aba^*b^*)$.
 - $\mathcal{L}((\lambda + \emptyset)^*)$.
 - $\mathcal{L}(aa^*bb^*aa^*)$.

- (g). $\mathcal{L}(ab + (\emptyset + a)^*b)$.
- (h). $\mathcal{L}((a^*(b(bb)^*aa)^*)a)$.
- (i). $\mathcal{L}(((aa)^*)^*)$.
- (j). $\mathcal{L}((b+a)(\lambda^*)^*)$.
- (k). $\mathcal{L}(a^*(ba+ab))$.
- (l). $\mathcal{L}((bb+bab)^*(a+\lambda aa^*+ba))$.
- (m). $\mathcal{L}((b^*+b)^*(a^*+\lambda))$.
- (n). $\mathcal{L}(a^*+ba)$.
- (o). $\mathcal{L}(((\lambda)^*+\emptyset)^*)$.
- (p). $\mathcal{L}(a^*ba^*ba^*)$.
- (q). $\mathcal{L}(bab + (ab+a)b)$.
- (r). $\mathcal{L}(a(b)^*b^*)$.
- (s). $\mathcal{L}((b+a)(b+a))$.
36. Construa uma expressão regular cuja valoração seja exatamente a linguagem aceita pelo AFD da Figura 6.26.
37. Construa uma gramática regular à esquerda para cada uma das linguagens descrita no Exercício 3.
38. Construa uma gramática regular à direita para cada uma das linguagens descrita no Exercício 3.
39. Construa uma gramática regular à esquerda para cada uma das linguagens descrita no Exercício 20.
40. Construa uma gramática regular à direita para cada uma das linguagens descrita no Exercício 20.
41. Construa uma gramática regular que gera exatamente cada valoração a seguir.
- (a). $\mathcal{L}(a^*b+a)$.
- (b). $\mathcal{L}((ab+cd)^*)$.
- (c). $\mathcal{L}((a^*ab)^*+bc)$.
- (d). $\mathcal{L}((aa)^*(b+\lambda))$.
- (e). $\mathcal{L}((a^*)+b(aa)^*)$.
42. Construa uma gramática regular que gera exatamente a linguagem aceita pelo AFD da Figura 6.26.
43. Para as gramáticas descrita a seguir construa um AFN que aceita as linguagens geradas por tais gramáticas.
- (a). $G_1 = \langle \{S, A, B\}, \{a, b\}, S, P \rangle$ onde P é definido por,
- $$\begin{aligned} S &\triangleright aA \mid bB \mid aaS \mid bbS \\ A &\triangleright aA \mid \lambda \\ B &\triangleright bB \mid b \end{aligned}$$
- (b). $G_2 = \langle \{S, A, B\}, \{a, b\}, S, P \rangle$ onde P é definido por,
- $$\begin{aligned} S &\triangleright abA \\ A &\triangleright bab \\ B &\triangleright aA \mid bb \end{aligned}$$
- (c). $G_3 = \langle \{S, A, B\}, \{a, b\}, S, P \rangle$ onde P é definido por,
- $$\begin{aligned} S &\triangleright aA \mid bS \mid \lambda \\ A &\triangleright aB \mid bS \mid \lambda \\ B &\triangleright aaS \mid bS \mid \lambda \end{aligned}$$

(d). $G_4 = \langle \{S, A\}, \{a, b\}, S, P \rangle$ onde P é definido por,

$$S \triangleright aaB \mid b$$

$$A \triangleright bbS$$

(e). $G_5 = \langle \{S, A, B\}, \{a, b\}, S, P \rangle$ onde P é definido por,

$$S \triangleright aaaA \mid bbB$$

$$A \triangleright abaA \mid S$$

$$B \triangleright bbS \mid aA \mid bb$$

44. Para cada gramática do Exercício 43 esboce uma gramática regular unitária à esquerda equivalente.

45. Esboce uma gramática regular que gere a linguagem aceita pelo AFD da Figura 6.25.

Capítulo Linguagens Livres do Contexto

Tópicos

- ☐ Gramática Livres do Contexto
- ☐ Simplificação e Formas Normais de GLC
- ☐ Sobre Algoritmos de Pertinência em GLC
- ☐ GLC e Linguagens de Programação
- ☐ Autômato de Pilha
- ☐ Álgebras das LLC
- ☐ Problemas de Decisão das LLC
- ☐ Questionário

No capítulo passado foram apresentadas três diferentes formalismos para a classe das linguagens regulares, a saber, o formalismo operacional (os autômatos), o formalismo denotacional (as expressões regulares) e por fim o formalismo gerador ou axiomático (as gramáticas regulares). Agora este manuscrito irá continuar o estudo das linguagens formais apresentando a classe das linguagens livres do contexto, como antes serão apresentados diferentes formalismo para tal classe de linguagens.

7.1 Gramática Livres do Contexto

O estudo das Linguagens Livres do Contexto será iniciado aqui pela apresentação de seu formalismo gerador, ou seja, será primeiro apresentado a noção de Gramática Livre do Contexto.

Definição 7.1 (Gramática Livre do Contexto)

[12] Uma Gramática Livre do Contexto, ou simplesmente GLC, é uma gramática formal $G = \langle V, \Sigma, S, P \rangle$ onde todo $\alpha \triangleright \beta \in P$ é tal que $\alpha \in V$ e $\beta \in (V \cup \Sigma)^*$.

Exemplo 7.1 A estrutura $G = \langle \{A, B, C\}, \{0, 1\}, A, P \rangle$ onde P é definido pela regras:

$$\begin{aligned} A &\triangleright A0110BC \mid 0110C \mid AAB0110CC \mid \lambda \\ B &\triangleright 01B10 \mid 0110 \\ C &\triangleright 01C \mid \lambda \end{aligned}$$

é uma GLC.

Obviamente o leitor atento pode notar que a Definição 7.1 garante que toda Gramática Regular é um caso particular de GLC, porém o inverso não é verdadeiro, basta notar que a gramática apresentada no Exemplo 7.1, fere a definição de gramática regular apresentada na Seção 6.8 do capítulo passado. Agora utilizando a Definição 5.19 pode-se formalizar o conceito de Linguagem Livre do Contexto.

Definição 7.2 (Linguagem Livre do Contexto)

A linguagem L gerada por uma GLC G , ou seja, $L = \mathcal{L}(G)$, será chamada de Linguagem Livre do Contexto, ou simplesmente LLC.

Como para os casos das linguagens reconhecidas por AFD, para provar que uma determinada GLC G gera uma linguagem L é necessário demonstrar o seguinte resultado $w \in L \iff w \in \mathcal{L}(G)$.

Exemplo 7.2 A linguagem $L = \{a^i b^i \mid i > 0\}$ é gerado pela GLC $G = \langle \{S\}, \{a, b, c\}, S, P \rangle$ onde P é formado pelas regras,

$$S \triangleright aSb \mid ab$$

Demonstração (\Rightarrow) Suponha que $w \in L$ assim $w = a^i b^i$ para $i > 0$, agora por indução sobre a quantidade n de derivações em G será mostrado que toda forma sentencial gerada por G é da forma $a^n S^j b^n$ com $j \in \{0, 1\}$.

Base da indução: Com $n = 1$, é trivial pelas regras em P .

Hipótese indutiva (HI): Assuma que com n derivações tal que $n \geq 0$ a forma sentencial $a^n S^j b^n$ é gerada pela GLC G , ou seja, $S \gg^n a^n S^j b^n$ com $j \in \{0, 1\}$.

Passo indutivo: Agora dado $S \gg^{n+1} w'$, ou seja, w' é derivada de S com $n + 1$ derivações, assim por definição tem-se que existe w'' tal que $S \gg^n w'' \gg w'$, mas por (HI) tem-se que $w'' = a^n S^j b^n$ com $n \geq 0$ e $j \in \{0, 1\}$, desde que, w' é gerada de w'' é claro que $j = 1$, ou seja, $w'' = a^n S b^n$, consequentemente, $w' = a^{n+1} S b^{n+1}$ ou $w' = a^{n+1} b^{n+1}$ e, portanto, w' é da forma $a^{n+1} S^j b^{n+1}$ com $j \in \{0, 1\}$.

Agora por hipótese tinha-se que $w = a^i b^i$ para $i > 0$, ou seja, $w = a^i S^0 b^i$, dessa forma pela indução acima w é uma forma sentencial gerada por G , e como $w \in \Sigma^*$ tem-se por definição que $w \in \mathcal{L}(G)$. (\Rightarrow) Suponha que $w \in \mathcal{L}(G)$, agora será mostrado por indução sobre a quantidade n de derivações que todo w gerado por G estará em L .

Base da indução: Com $n = 1$, ou seja, como uma única derivação, pelo fato de que $w \in \Sigma^*$ e pelas regras em P tem-se obrigatoriamente que $w = ab$ e, portanto, $w = a^1 b^1$, consequentemente, $w \in L$.

Hipótese indutiva (HI): Assuma que para todo $S \gg^n w$ tal que $n \geq 1$, tem-se que $w \in L$.

Passo indutivo: Agora dado $S \gg^{n+1} w$, ou seja, w é derivado em G com $n + 1$ derivações de forma que $n \geq 1$. Desde que $n \geq 1$ tem-se que $n + 1 > 1$, assim a palavra ab não pode ser w uma vez que são usadas pelo menos duas derivações para gerar w . Assim a derivação de w deverá ter sido iniciado usando a regra $S \triangleright aSb$ e, portanto, $w = aw'b$ em que $S \gg^n w'$, mas pela hipótese indutiva $w' \in L$, logo $w' = a^i b^i$ com $i > 0$, consequentemente $w = aa^i b^i b = a^{i+1} b^{i+1}$, logo $w \in L$. ■

Note que para a prova mostrada no Exemplo 7.2 foi-se usada indução sobre o tamanho das derivações, isto não é a única forma de se proceder para provar que uma gramática gera uma determinada linguagem, de fato como apresentado em [33] pode-se usar a ideia de indução sobre as árvores de derivações (apresentadas mais a frente), neste manuscrito não será apresentado tal estratégia, porém fica a referência para interessados no tema.

Nas GLC que não são lineares, ou seja, as gramática em que as regras em P podem conter duas ou mais variáveis do lado direito do símbolo \triangleright , é permitido a escolha sobre qual variável ser derivada primeiramente, para ilustrar isso considere uma forma sentencial $0A10BC$ e as regras $A \triangleright 00$, $B \triangleright \lambda$ e $C \triangleright 1$, note que dependendo de qual variável é escolhida para ser reescrita a próxima forma sentencial pode assumir três formas diferentes. Apresentado esta ideia pode-se agora formalizar o conceito de reescrita **mais à esquerda** e **mais à direita**.

Definição 7.3 (Tipos de Derivação)

Dado uma GLC $G = \langle V, \Sigma, S, P \rangle$ e seja $w_1 A_1 w_2 \cdots w_n A_n w_{n+1}$ uma forma sentencial derivável em G tal que $w_i \in \Sigma^*$ e $A_j \in V$ para todo $1 \leq i \leq n + 1$ e $1 \leq j \leq n$. Uma derivação é dita **mais à esquerda** (à direita) se ela é da forma $w_1 A_1 w_2 \cdots w_n A_n w_{n+1} \gg w_1 w' w_2 \cdots w_n A_n w_{n+1}$ ($w_1 A_1 w_2 \cdots w_n A_n w_{n+1} \gg w_1 A_1 w_2 \cdots w_n w' w_{n+1}$) com $w' \in (V \cup \Sigma)^*$. ♣

Exemplo 7.3 Considere a GLC $G = \langle \{S, A, B\}, \{0, 1\}, S, P \rangle$ em que P é formado pelas regras,

$$S \triangleright 0AB$$

$$A \triangleright 11B1$$

$$B \triangleright A \mid \lambda$$

claramente a palavra $011B1A$ é um forma sentencial derivável em G , agora note que:

$$011B1A \gg 011A1A$$

$$011B1A \gg 0111A$$

são ambas derivações mais à esquerda, e

$$011B1A \gg 011B111B1$$

é uma derivação mais à direita.

Definição 7.4 (Tipo de Geração)

Dado uma GLC $G = \langle V, \Sigma, S, P \rangle$, uma derivação $S \gg^ w$ é dita ser uma geração mais à esquerda (à direita) sempre que $w \in \Sigma^*$ e toda derivação realizada for mais à esquerda (à direita).*

Exemplo 7.4 Considere a gramática do Exemplo 7.3 a derivação,

$$S \gg 0AB \gg 011B1B \gg 0111B \gg 0111$$

é claramente uma geração mais à esquerda, já a derivação

$$S \gg 0AB \gg 0A \gg 011B1 \gg 0111$$

é notavelmente uma geração mais à direita.

Observação: Resumindo dado uma GLC $G = \langle V, \Sigma, S, P \rangle$ uma geração é obrigatoriamente uma derivação $S \gg w$ em que $w \in \Sigma^*$, ou seja, uma derivação que gera (ou produz) uma palavra sobre o alfabeto Σ .

Além da noção de geração, para visualizar o passo a passo da “construção” de uma palavra $w \in \Sigma^*$ em uma GLC G é comum usar a ideia de árvore de derivação¹.

Uma árvore de derivação nada mais é do que uma forma visual estruturada em uma árvore² que representar a construção de uma palavra, em que cada nível da árvore representa é gerada pela aplicação de uma ou mais regras de reescrita, neste sentido as folhas da árvore contém os símbolos da palavra w .

Definição 7.5 (Árvore de Derivação)

Seja $G = \langle V, \Sigma, S, P \rangle$ uma GLC e dado $w \in \mathcal{L}(G)$, uma árvore de derivação para w é construída seguindo as seguintes regras:

1. *O símbolo S é a raiz da árvore.*
2. *Todo símbolo $a \in \Sigma \cup \{\lambda\}$ é o rótulo de uma folha.*
3. *Todo símbolo $A \in V$ é o rótulo de uma raiz de uma sub-árvore^a.*
4. *Se A é uma raiz e seus “filhos” são x_1, \dots, x_n com $x_i \in V \cup \Sigma \cup \{\lambda\}$ para $1 \leq i \leq n$, então existe em P uma regra da forma $A \triangleright x_1 \cdots x_n$.*
5. *Todo folha rotulada por λ de uma raiz A será filho único, ou seja, em P existe uma regra da forma $A \triangleright \lambda$.*

^aAqui sub-árvore e no mesmo sentido que o leitor já deve ter visto em estrutura de dados.

¹Na literatura como no livro [12] também é comum encontrar a nomenclatura árvore ordenada.

²As árvores de derivação são representações cuja a forma similar a construção das árvores n -árias da área de estrutura de dados [60].

Exemplo 7.5 Tem-se que a derivação $S \Rightarrow 0AB \Rightarrow 011B1B \Rightarrow 0111B \Rightarrow 0111$ apresentada no Exemplo 7.4 é representada pela árvore esboçada na Figura 7.1 a seguir.

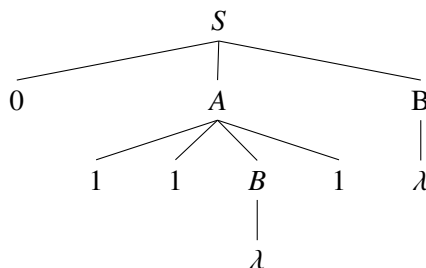


Figura 7.1: Árvore para a derivação $S \Rightarrow 0AB \Rightarrow 011B1B \Rightarrow 0111B \Rightarrow 0111$.

Exemplo 7.6 Tem-se que a derivação $S \Rightarrow 0AB \Rightarrow 0A \Rightarrow 011B1 \Rightarrow 0111$ apresentada no Exemplo 7.4 é representada pela árvore esboçada na Figura 7.2 a seguir.

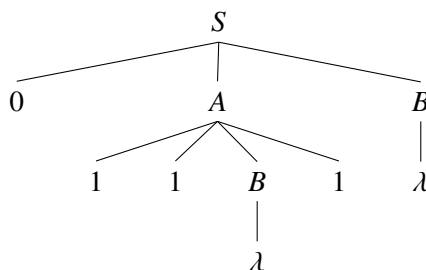


Figura 7.2: Árvore para a derivação $S \Rightarrow 0AB \Rightarrow 0A \Rightarrow 011B1 \Rightarrow 0111$.

Observação: Os Exemplos 7.5 e 7.6 mostram que diferentes derivações podem ser representadas pela mesma árvore, isso acontece devido ao fato de que a árvore de derivação apenas esboça o processo de formação total (ou final), e não o comportamento (ou momento) parcial das derivação.



Nota: Em algumas obras tais como [12] as folhas e raízes nas árvores de derivação são gravados com círculos, neste manuscrito isso não será feito para não confundir o leitor iniciante com a representação visual dos autômatos finitos.

Definição 7.6 (Ambiguidade)

Dado uma GLC $G = \langle V, \Sigma, S, P \rangle$ uma palavra $w \in \mathcal{L}(G)$ é dita ser ambígua sempre que existe duas gerações diferentes para w .



A Definição 7.6 pode ser reinterpretada da seguinte forma: uma palavra w é ambígua com relação a uma GLC G se existem duas gerações mais à esquerda (à direita) para tal palavra.

Definição 7.7 (Linguagem Ambígua)

Uma linguagem L é dita ser ambígua se existe uma GLC G e uma palavra w tal que $L = \mathcal{L}(G)$ e w seja uma palavra ambígua em G . L será dita inerentemente ambígua se existe w para todo GLC G tal que w seja uma palavra ambígua em G e $L = \mathcal{L}(G)$.



Exemplo 7.7 Considere a GLC $G = \langle \{S\}, \{a, b\}, S, P \rangle$ onde P é o formado pelas seguintes regras:

$$S \triangleright aSb \mid SS \mid \lambda$$

a palavra $aabb$ é ambígua, pois existem as duas árvores de derivação apresentadas na Figura 7.3, assim tem-se

que $\mathcal{L}(G)$ é uma linguagem ambígua.

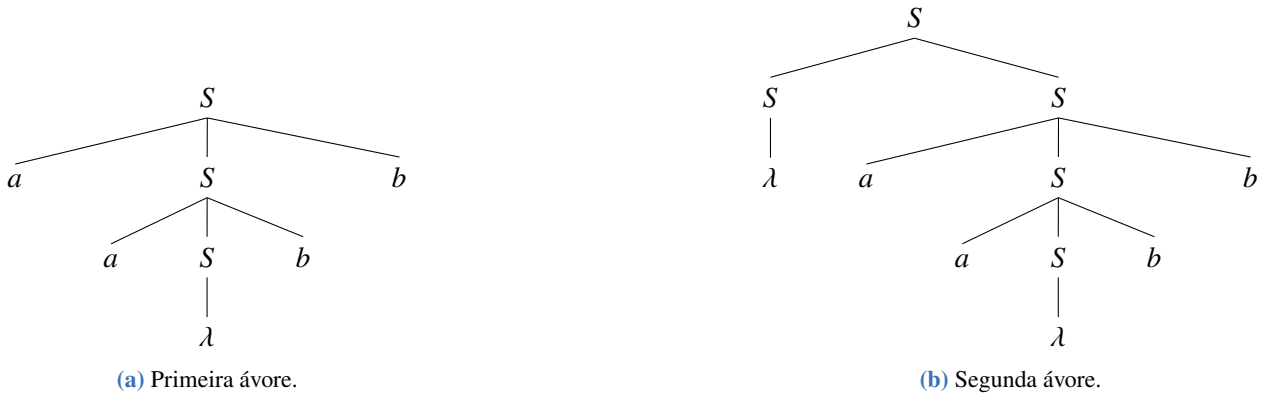


Figura 7.3: As árvores de derivação para a palavra $aabb$

Ambiguidade é uma característica comum encontrada nas linguagem naturais [12] e em tais linguagens essa característica é tolerada, em contrapartida, as linguagens de programação não toleram muito bem o aspecto da ambiguidade assim em geral a mesma deve ser sempre que possível evitada ou eliminada. Uma estratégia comum para a eliminação da ambiguidade em linguagens de programação é estabelecer prioridade na geração de certos símbolos da linguagens [3, 12].

7.2 Simplificação de Gramática Livres do Contexto

Antes de apresentar as regras de simplificação e alguns resultados sobre as mesmas é conveniente discutir alguns aspectos sobre as GLC. Antes de tudo lembre que na definição de GLC (Definição 7.1) não existe qualquer restrição para a forma das palavras encontradas à direita das regras de reescrita. Assim podem haver n variáveis, recursão, geração de λ e etc. Algumas desta, entretanto, podem ser removidas ou alteradas de forma que a gramática possa ser mais simples de ser utilizada, e será isto que será estudado nesta seção, ou seja, aqui será estudado diversas estratégias para transformar uma gramática qualquer G em uma gramática G' com alguns restrições, mas que preserva a linguagem gerada, isto é, $\mathcal{L}(G) = \mathcal{L}(G')$.

Observação: Assim como em [12] as gramáticas que serão consideradas nesta seção não são capazes de gerar a palavra vazia, ou seja, $S \not\Rightarrow \lambda$ ou ainda $\lambda \notin \mathcal{L}(G)$. Entretanto isso não irá diminuir em nada a força dos resultados que serão obtidos aqui.

Definição 7.8 (Regra da substituição)

Seja $G = \langle V, \Sigma, S, P \rangle$ uma GLC e seja $A, B \in V$ tal que $A \neq B$ de forma que em P existem as produções:

$$A \triangleright x_1 B x_2$$

e

$$B \triangleright y_1 \mid y_2 \mid \cdots \mid y_n$$

a regra de substituição consiste em para cada a regra $A \triangleright x_1 A x_2$ em P adicionar as regras:

$$A \triangleright x_1 y_1 x_2 \mid x_1 y_2 x_2 \mid \cdots \mid x_1 y_n x_2 \mid$$

.



Note que a regra de substituição de fato só altera a forma do conjunto P gerando um novo conjunto P' , e

pela própria regra é fácil notar que vale a seguinte desigualdade $\#P \leq \#P'$.

Teorema 7.1

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então a gramática G' gerada a partir da regra de substituição é tal que $\mathcal{L}(G) = \mathcal{L}(G')$.

Demonstração Suponha $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, agora seja $G' = \langle V, \Sigma, S, P' \rangle$ a GLC gerada a partir da regra de substituição aplicada a G . Agora assuma que $w \in \mathcal{L}(G)$, ou seja, $S \gg_G^* w$, dessa forma há dois casos para serem considerados:

- (1) Se durante a derivação $S \gg_G^* w$ não forem usadas regras da forma $A \triangleright x_1 B x_2$, então obviamente essa mesma derivação pode ser replicada em todos os seu detalhes na gramática G' , uma vez que, todas as regras que não são da forma $A \triangleright x_1 B x_2$ estão também em P' , assim $S \gg_{G'}^* w$, consequentemente, $w \in \mathcal{L}(G')$.
- (2) Agora sem perda de generalidade assuma que pelo menos uma regra da forma $A \triangleright x_1 B x_2$ seguida da regra $B \triangleright y_1$ aparecem na derivação $S \gg_G^* w$, portanto, tem-se que:

$$S \gg_G^* k_1 A k_2 \gg_G k_1 x_1 B x_2 k_2 \gg_G k_1 x_1 y_1 x_2 k_2 \gg_G^* w$$

com $k_1, k_2, x_1, x_2, y_1 \in (V \cup \Sigma)^*$. Assim pela construção de G' pode-se então realizar a seguinte derivação:

$$S \gg_{G'}^* k_1 A k_2 \gg_{G'} k_1 x_1 y_1 x_2 k_2 \gg_{G'}^* w$$

e assim tem-se que, $w \in \mathcal{L}(G')$.

Logo pelos casos (1) e (2) acima tem-se que $\mathcal{L}(G) \subseteq \mathcal{L}(G')$. Agora mostrar que $\mathcal{L}(G') \subseteq \mathcal{L}(G)$ é trivial e não será feito aqui, e desde que $\mathcal{L}(G) \subseteq \mathcal{L}(G')$ e $\mathcal{L}(G') \subseteq \mathcal{L}(G)$ por definição tem-se que $\mathcal{L}(G) = \mathcal{L}(G')$ o que completa a prova. ■

Como pode ser visto no Teorema 7.1 e como discutido em [12] a regra de substituição pode até aumentar o número de regras de reescrita, porém, a mesma também permite que seja realizadas geração mais rápidas, isto é, que aconteçam derivações de palavras $w \in \Sigma^*$ com menos passos.

Exemplo 7.8 Dado a GLC $G = \langle \{S, A, B, C\}, \{0, 1\}, S, P \rangle$ com P definido pelas regras:

$$S \triangleright A001 \mid B100$$

$$A \triangleright 01 \mid 01B$$

$$B \triangleright 1 \mid 0 \mid \lambda$$

aplicando a regra de substituição em tal gramática será gerada na gramática $G' = \langle \{S, A, B, C\}, \{0, 1\}, S, P' \rangle$ em que P' corresponde ao conjunto formado pela seguintes regras:

$$S \triangleright A001 \mid B100 \mid 01001 \mid 01B001 \mid 1100 \mid 0100 \mid 100011001 \mid 010001 \mid 01001$$

$$A \triangleright 01 \mid 01B \mid 011 \mid 010$$

$$B \triangleright 1 \mid 0 \mid \lambda$$

Para prosseguir com o estudo das regras de simplificação antes é necessário formalizar o conceito de variável recursiva à esquerda nas GLC.

Definição 7.9 (Regra Recursiva à Esquerda)

Seja $G = \langle V, \Sigma, S, P \rangle$ uma GLC, uma variável A é dita recursiva à esquerda se existe pelo menos uma regra $A \triangleright Ax \in P$ com $A \in V$ e $x \in (V \cup \Sigma)^*$.

Agora é possível formalizar então a próxima regra de simplificação de GLC, sendo esta próxima regra

nomeada como **Regra de Remoção da Recursividade Esquerda**.

Definição 7.10 (Regra de Remoção da Recursividade Esquerda)


Seja $G = \langle V, \Sigma, S, P \rangle$ uma GLC com variáveis recursivas à esquerda. Então substitua para cada $A \in V$ as regras:

$$A \triangleright Ax_1 \mid \cdots \mid Ax_m \mid y_1 \mid \cdots \mid y_n$$

pelas regras:


$$A \triangleright y_1 \mid \cdots \mid y_n \mid y_1Z \mid \cdots \mid y_nZ$$


$$Z \triangleright x_1 \mid \cdots \mid x_m \mid x_1Z \mid \cdots \mid x_mZ$$

onde $x_1, \dots, x_m, y_1, \dots, y_n \in (V \cup \Sigma)^*$ e Z é uma variável nova que deve ser adicionada ao conjunto V . 

Fica claro pela definição da Regra de Remoção da Recursividade Esquerda que após sua aplicação será gerada uma nova gramática com mais variáveis que a gramática original terá uma quantidade igual ou superior de regras de reescritas.

Teorema 7.2

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então a gramática G' gerada a partir da regra de remoção da recursividade esquerda é tal que $\mathcal{L}(G) = \mathcal{L}(G')$. 

Demonstração Ficarà como exercício ao leitor. 

Exemplo 7.9 Dado a GLC $G = \langle \{S, A\}, \{a, b, c\}, S, P \rangle$ com P definido pelas regras:

$$S \triangleright aSc \mid aAc$$

$$A \triangleright Ab \mid b$$

Usando a regra de remoção da recursividade esquerda é obtida a GLC $G = \langle \{S, A, B\}, \{a, b, c\}, S, P' \rangle$ onde P' é definido por:


$$S \triangleright aSc \mid aAc$$

$$A \triangleright b \mid bB$$

$$B \triangleright b \mid bB$$

Agora para continuar com o estudo das regras de simplificação será necessário formalizar três conceitos-chaves, o primeiro conceito é a noção de relação (ou grafo) de dependências, a saber tais relações (ou grafos) são ferramentas advindas da teoria dos grafos e frequentemente usada em muitos campos do conhecimento para modelar sistemas complexos [12], a seguir este conceito será apresentado de forma rigorosa com respeito as GLC.

Definição 7.11 (Relação de dependência)

Seja $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, a relação de dependência DEP é construída como: $(A, B) \in DEP$ se, e somente se, $A \triangleright xBy \in P$ com $x, y \in (V \cup \Sigma)^*$ e $A \neq B$. 

O leitor pode considerar a relação DEP como sendo uma relação de necessidade, em que $(A, B) \in DEP$ pode ser semanticamente interpretado como, para escrever B é necessário que antes existam um A .

Observação: Para fins de notação, será usado o rótulo \widehat{DEP} para denotar o fecho transitivo da relação DEP .


Exemplo 7.10 Considere a GLC $G = \langle \{A, B, C, D, E\}, \{0, 1\}, A, P \rangle$ onde P é o conjunto formado pelas regras:

$$\begin{aligned}
A &\triangleright 00A \mid 0B1 \mid 10AB \mid AD \\
B &\triangleright 1B01 \mid D01E \mid \lambda \\
C &\triangleright C01 \mid 00101 \mid 1 \mid \lambda \\
D &\triangleright 01E \mid 0E \\
E &\triangleright 1E0 \mid EE \mid D
\end{aligned}$$

Tem-se que \widehat{DEP} é o conjunto formado pelo pares (A, B) , (A, D) , (B, D) , (B, E) , (D, E) , e (E, D) .

O próximo conceito necessário é a noção de variável inacessível formalizado a seguir.

Definição 7.12 (Variável inacessível)

Seja $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, uma variável $A \neq S$ é dita ser inacessível sempre que $(S, A) \notin \widehat{DEP}$. 


Exemplo 7.11 Considere a GLC apresentada no Exemplo 7.10, em tal gramática é fácil notar que a variável C é inacessível.

Lema 7.1

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então existe uma GLC sem variáveis inacessíveis $G' = \langle V', \Sigma, S, P' \rangle$ tal que $\mathcal{L}(G) = \mathcal{L}(G')$. 


Demonstração Suponha que $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, assim pode-se construir uma nova GLC $G' = \langle V', \Sigma, S, P' \rangle$ onde:

- $V' = V - \{A \in V \mid A \text{ é uma variável inacessível}\}$ e
- $P' = \{A \triangleright xBy \in P \mid A, B \in V', x, y \in (V' \cup \Sigma)^*\}$.

Agora é claro que G' possui apenas variáveis acessíveis. Note agora que para toda $S \gg_G w$ tem-se que apenas regras com variáveis acessíveis são utilizadas em tal derivação, além disso, é claro pela construção de G' que tais regras estarão em P' , conseqüentemente $S \gg_{G'} w$, assim $\mathcal{L}(G) \subseteq \mathcal{L}(G')$. Por outro lado, assuma por absurdo que $\mathcal{L}(G') \not\subseteq \mathcal{L}(G)$, assim existe um $w \in \mathcal{L}(G')$ tal que $w \notin \mathcal{L}(G)$, desde que $w \in \mathcal{L}(G')$ tem-se que $S \gg_{G'} w$, mas pela construção de G' é óbvio que toda regra usada na derivação $S \gg_{G'} w$ também estão presente em P e, portanto, $S \gg_G w$ o que contradiz a hipótese de que $\mathcal{L}(G') \not\subseteq \mathcal{L}(G)$, conseqüentemente, tem-se $\mathcal{L}(G') \subseteq \mathcal{L}(G)$. Desde que, $\mathcal{L}(G) \subseteq \mathcal{L}(G')$ e $\mathcal{L}(G') \subseteq \mathcal{L}(G)$, pela definição de igualdade entre conjunto tem-se que $\mathcal{L}(G) = \mathcal{L}(G')$. 

O terceiro e último conceito necessário para continuar com o estudo das regras de simplificação é a ideia de variável descartável em uma GLC, de forma direta uma variável descartável é aquela incapaz de derivar uma palavra sobre o alfabeto Σ , em notação formal tem-se a definição a seguir.

Definição 7.13 (Variável descartável)

Seja $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, uma variável A é dita descartável sempre que para todo $w \in \Sigma^*$ tem-se que $A \not\Rightarrow^* w$. 

Exemplo 7.12 Considere a GLC apresentada no Exemplo 7.10, em tal gramática as variáveis D e E são ambas descartáveis.

O leitor um pouco mais experiente em programação pode facilmente notar que uma variável descartável é aquela que é a raiz de uma árvore infinita, isto é, uma árvore que continua crescendo infinitamente, outra forma de interpretar uma variável descartável é como sendo uma variável que produz um *loop* infinito no processo

de derivação, ou seja, a partir de tal variável nunca é possível derivar uma palavra sobre o alfabeto base da gramática.

Lema 7.2

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então existe uma GLC sem variáveis descartáveis $G' = \langle V', \Sigma, S, P' \rangle$ tal que $\mathcal{L}(G) = \mathcal{L}(G')$.

Demonstração Suponha que $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, assim pode-se construir uma nova GLC $G' = \langle V', \Sigma, S, P' \rangle$ onde:

- $V' = V - \{A \in V \mid A \text{ é uma variável descartável}\}$ e
- $P' = \{A \triangleright xBy \in P \mid A, B \in V', x, y \in (V' \cup \Sigma)^*\}$.

Agora é claro que G' não possui variáveis descartáveis. A argumentação para provar $\mathcal{L}(G) = \mathcal{L}(G')$ é similar a demonstração do Lema 7.1. ■

Definição 7.14 (Regra de remoção de variáveis inúteis)

Dado uma GLC $G = \langle V, \Sigma, S, P \rangle$ será gerada uma gramática G' sem variáveis inúteis, após a execução dos seguintes passos.

1. Remova todas as variáveis Descartáveis e assim gere uma gramática G_1 .
2. Remova da gramática G_1 todas as variáveis Inacessíveis, e assim gere a gramática G' .

O termo inútil diz respeito ao fato de uma variável A não ser capaz de derivar uma palavra $w \in \Sigma^*$, ou seja, $A \not\Rightarrow_G w$.

Teorema 7.3

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então a GLC G' gerada pela regra de remoção de variáveis inúteis é tal que $\mathcal{L}(G) = \mathcal{L}(G')$.

Demonstração Direto dos Lemas 7.1 e 7.2. ■

O exemplo a seguir ilustra a aplicação da regra de remoção de variáveis inúteis.

Exemplo 7.13 Considere a GLC $G = \langle \{S, T, V, X, Y, Z\}, \{0, 1\}, S, P \rangle$ onde P é o conjunto com as seguintes produções:

$$\begin{aligned} S &\triangleright 01S \mid SX0 \mid YZ \mid X0Z11 \\ T &\triangleright \lambda \\ X &\triangleright X01 \mid Y0X \\ Y &\triangleright S01 \mid YY \mid XY \mid \lambda \\ Z &\triangleright S01 \mid SS \end{aligned}$$

Removendo as variáveis descartáveis (neste caso existe apenas variável X a ser removida) é gerado a GLC $G_1 = \langle \{S, T, V, Y, Z\}, \{0, 1\}, S, P_1 \rangle$ com P_1 sendo:

$$\begin{aligned} S &\triangleright 01S \mid YZ \\ T &\triangleright \lambda \\ Y &\triangleright S01 \mid YY \mid \lambda \\ Z &\triangleright S01 \mid SS \end{aligned}$$

Agora trabalhando considerando o conjunto P_1 pode-se remover as variáveis inacessíveis, ou seja, são removidas

as variáveis T, V e assim tem-se como resultado final da remoção de variáveis inúteis é gerada a GLC $G' = \langle \{S, Y, Z\}, \{0, 1\}, S, P' \rangle$ onde P' é formado por:

$$\begin{aligned} S &\triangleright 01S \mid YZ \\ Y &\triangleright S01 \mid YY \mid \lambda \\ Z &\triangleright S01 \mid SS \end{aligned}$$

Definição 7.15

Seja $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, uma regra da forma $A \triangleright \lambda$ é chamada de λ -produção.

O próximo resultado estabelece que em GLC a existência de λ -produções não aumenta o poder gerativo das gramáticas.

Teorema 7.4

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então existe uma GLC G' sem λ -produções tal que $\mathcal{L}(G) = \mathcal{L}(G')$.

Demonstração Suponha que $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, agora defina os conjuntos

$$\begin{aligned} P_1 &= P - \{A \triangleright \lambda \mid A \in V\} \\ V_\lambda &= \{A \in V \mid A \triangleright \lambda\} \end{aligned}$$

Pode-se agora construir um novo conjunto de regras P_2 da seguinte forma, para cada uma das regras

$$A \triangleright w_1 A_1 w_2 A_2 w_3 \cdots w_{n-1} A_{n-1} w_n A_n w_{n+1} \in P_1$$

com $w_i \in (V \cup \Sigma)^*$, $A_j \in V_\lambda$ sendo $1 \leq i \leq n+1$ e $1 \leq j \leq n$ estarão em P_2 as regras:

$$\begin{aligned} A &\triangleright w_1 w_2 A_2 w_3 \cdots w_{n-1} A_{n-1} w_n A_n w_{n+1} \\ A &\triangleright w_1 A_1 w_2 w_3 \cdots w_{n-1} A_{n-1} w_n A_n w_{n+1} \\ &\vdots \\ A &\triangleright w_1 A_1 w_2 A_2 w_3 \cdots w_{n-1} w_n A_n w_{n+1} \\ A &\triangleright w_1 A_1 w_2 A_2 w_3 \cdots w_{n-1} A_{n-1} w_n w_{n+1} \\ A &\triangleright w_1 w_2 w_3 \cdots w_{n-1} w_n w_{n+1} \end{aligned}$$

Finalmente defina a GLC $G' = \langle V, \Sigma, S, P' \rangle$ em que $P' = P_1 \cup P_2$, é claro por sua construção que P' não possui λ -produções. Agora assuma que $w \in \mathcal{L}(G)$, ou seja, $S \gg_G^* w$, dessa forma há dois casos para serem considerados:

- (1) Se durante a derivação $S \gg_G^* w$ não forem usadas regras da forma $A \triangleright \lambda$, então obviamente essa mesma derivação pode ser replicada em todos os seu detalhes na gramática G' , uma vez que, todas as regras que não são da forma $A \triangleright \lambda$ estão também em P' , assim $S \gg_{G'}^* w$, consequentemente, $w \in \mathcal{L}(G')$.
- (2) Agora sem perda de generalidade assuma que pelo menos uma regra da forma $A \triangleright x_1 B x_2$ seguida da regra $B \triangleright \lambda$ aparecem na derivação $S \gg_G^* w$, portanto, tem-se que:

$$S \gg_G^* k_1 A k_2 \gg_G k_1 x_1 B x_2 k_2 \gg_G k_1 x_1 x_2 k_2 \gg_G^* w$$

com $k_1, k_2, x_1, x_2, \in (V \cup \Sigma)^*$. Assim pela construção de G' existe uma regra da forma $A \triangleright x_1 x_2$ e, portanto, em G' é possível realizar a seguinte derivação

$$S \gg_G^* k_1 A k_2 \gg_{G'} k_1 x_1 x_2 k_2 \gg_G^* w$$

e assim tem-se que $w \in \mathcal{L}(G')$.

Assim pelos casos (1) e (2) mostrados acima tem-se que $\mathcal{L}(G) \subseteq \mathcal{L}(G')$. Por outro lado, mostrar que $\mathcal{L}(G') \subseteq \mathcal{L}(G)$ é trivial e ficará como exercício ao leitor. Desde que $\mathcal{L}(G) \subseteq \mathcal{L}(G')$ e $\mathcal{L}(G') \subseteq \mathcal{L}(G)$ por definição de igualdade de conjuntos tem-se que $\mathcal{L}(G) = \mathcal{L}(G')$. ■

Exemplo 7.14 Dado a gramática $G = \langle \{S, X, Y, Z\}, \{a, b, c\}, S, P \rangle$ com P sendo o conjunto com as seguintes regras,

$$\begin{aligned} S &\triangleright Xca \mid bYa \mid aZXcb \\ X &\triangleright bbXY \mid ZacYb \mid abYcZa \\ Y &\triangleright cYcaYX \mid \lambda \\ Z &\triangleright abc \mid \lambda \end{aligned}$$

utilizando a estratégia usada na prova do Teorema 7.4, tem-se o seguinte conjunto P_1 de regras,

$$\begin{aligned} S &\triangleright Xca \mid bYa \mid aZXcb \\ X &\triangleright bbXY \mid ZacYb \mid abYcZa \\ Y &\triangleright cYcaYX \\ Z &\triangleright abc \end{aligned}$$

já o conjunto P_2 corresponde ao seguinte conjunto,


$$\begin{aligned} S &\triangleright Xca \mid ba \mid aXcb \\ X &\triangleright bbX \mid acYb \mid Zacb \mid acb \mid abYcZa \mid abcZa \mid abYca \mid abca \\ Y &\triangleright cYcaYX \mid ccaYX \mid cYcaX \mid ccaX \\ Z &\triangleright abc \end{aligned}$$

por fim, é construído a gramática $G = \langle \{S, X, Y, Z\}, \{a, b, c\}, S, P' \rangle$ onde P' é o conjunto com as seguintes regras,


$$\begin{aligned} S &\triangleright Xca \mid bYa \mid aZXcb \mid ba \mid aXcb \\ X &\triangleright bbXY \mid ZacYb \mid abYcZa \mid bbX \mid acYb \mid Zacb \mid acb \mid abcZa \mid abYca \mid abca \\ Y &\triangleright cYcaYX \mid ccaYX \mid cYcaX \mid ccaX \\ Z &\triangleright abc \end{aligned}$$

Entre os diferentes tipo de regras em uma gramática, existe o tipo chamado de regras unitárias definidas a seguir, tais regras tem a natureza de não gerar qualquer símbolo do alfabeto, e essa natureza muitas vezes não é desejável como dito em [12], a seguir será mostrado que tais regras não aumentam o poder gerativo das GLC.

Definição 7.16 (Regras unitárias)

Seja $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, um elemento de P é chamado de regra unitária sempre que tal elemento for da forma $A \triangleright B$ com $A, B \in V$. 

Teorema 7.5

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC com regras unitárias, então existe uma GLC $G' = \langle V, \Sigma, S, P' \rangle$ sem regras unitárias tal que $\mathcal{L}(G) = \mathcal{L}(G')$. 

Demonstração Suponha que $G = \langle V, \Sigma, S, P \rangle$ é uma GLC com regras unitárias, assim construa o seguinte conjunto de regras,

$$P_1 = \{A \triangleright xBy \in P \mid |x| + |y| \geq 1, B \in V\}$$

agora construa o conjunto de regras,

$$P_2 = \{A \triangleright w \mid A \gg_G^* B, B \triangleright w \in P_1\}$$

é fácil notar que P_1 e P_2 não possuem regras unitárias. Por fim, defina a GLC $G' = \langle V, \Sigma, S, P' \rangle$ onde $P' = P_1 \cup P_2$. A argumentação para provar $\mathcal{L}(G) = \mathcal{L}(G')$ é similar a demonstração do Teorema 7.4. ■

A seguir para melhor fixação do conhecimento o leitor encontrará no próximo exemplo o uso da construção esboçada na prova do Teorema 7.5.

Exemplo 7.15 Dado a gramática $G = \langle \{S, X, Y, Z\}, \{0, 1\}, S, P \rangle$ com P sendo o conjunto com as seguintes regras,

$$\begin{aligned} S &\triangleright 0X \\ X &\triangleright Y \mid 0X \\ Y &\triangleright Z \mid 1Y \mid YY \\ Z &\triangleright Y \mid 01 \end{aligned}$$

agora utilizando a estratégia apresentada na prova do Teorema 7.5 tem-se que P_1 é formado pelas seguintes regras,

$$\begin{aligned} S &\triangleright 0X \\ X &\triangleright 0X \\ Y &\triangleright 1Y \mid YY \\ Z &\triangleright 01 \end{aligned}$$

e assim o conjunto P_2 é formado pelas regras

$$\begin{aligned} X &\triangleright 1Y \mid YY \mid 01 \\ Y &\triangleright 01 \\ Z &\triangleright 1Y \mid YY \end{aligned}$$

por fim, é construído a gramática $G = \langle \{S, X, Y, Z\}, \{a, b, c\}, S, P' \rangle$ onde P' é o conjunto com as seguintes regras,

$$\begin{aligned} S &\triangleright 0X \\ X &\triangleright 0X \mid 1Y \mid YY \mid 01 \\ Y &\triangleright 1Y \mid YY \mid 01 \\ Z &\triangleright 01 \mid 1Y \mid YY \end{aligned}$$

Agora que foram apresentadas as estratégia de simplificação de gramática este manuscrito pode prosseguir para a apresentação do conceito de normalização de gramática. Como explicado em [45], uma normalização consiste de estabelecer um ou mais padrões (ou restrições) que **todas as regras** de reescrita da gramática devem obedecer. As duas principais formas normais para as GLC são como dito em [12], a forma normal de Chomsky [18] e a forma normal de Greibach [27].

Vale ressaltar que nos dias atuais no estudo das linguagens formais é comum adotar uma definição reduzida³ da forma normal de Chomsky como podem ser visto em [12, 33, 45], assim seguindo o padrão atual do estudo das linguagens formais e autômatos esse manuscrito também irá adotar a versão reduzida.

³A diferença entre a forma reduzida e forma normal introduzida por Chomsky em [18], é que a versão original do paper permitia a existência da regra $S \triangleright \lambda$.

Definição 7.17 (Forma normal de Chomsky)

Uma GLC $G = \langle V, \Sigma, S, P \rangle$ está na forma normal Chomsky se para todo $\alpha \triangleright \beta \in P$ tem-se que $\beta = AB$ ou $\beta = a$ com $A, B \in V$ e $a \in \Sigma$.



Observação: É evidente pela própria definição que GLC na forma normal de Chomsky não possuem λ -produções e nem regras unitárias.

Antes de apresentar o algoritmo capaz de converter qualquer GLC em sua equivalente na forma normal de Chomsky é conveniente apresentar a definição e os resultados a seguir.

Definição 7.18 (Forma normal de pré-Chomsky)

Uma GLC $G = \langle V, \Sigma, S, P \rangle$ está na forma normal pré-Chomsky se para todo $\alpha \triangleright \beta \in P$ tem-se que $\beta = A_1 \cdots A_m$ ou $\beta = a$ com $A_i \in V$, $a \in \Sigma$ sendo $1 \leq i \leq m$.

**Lema 7.3**

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então existe uma GLC G' na forma normal pré-Chomsky tal que $\mathcal{L}(G) = \mathcal{L}(G')$.



Demonstração Assuma sem perda de generalidade que $G = \langle V, \Sigma, S, P \rangle$ é uma GLC sem λ -produções, assim toda regra $\alpha \triangleright \beta \in P$ é tal que $\beta = w_1 A_1 \cdots w_n A_n w_{n+1}$ com $w_i \in \Sigma^*$ e $A_j \in V \cup \{\lambda\}$ e $|w_1 A_1 \cdots w_n A_n w_{n+1}| \geq 1$ sendo que $1 \leq i \leq n+1$ e $1 \leq j \leq n$. Agora construa um novo conjunto de variáveis da forma $V_\Sigma = \{A_a \mid a \in \Sigma\}$, assim é claro a existência de isomorfismo $h : \Sigma^* \rightarrow V_\Sigma^*$ definido recursivamente para todo $w \in \Sigma^*$ e $a \in \Sigma$ como sendo:

$$\begin{aligned} h(\lambda) &= \lambda \\ h(wa) &= h(w)A_a \end{aligned}$$

agora defina uma nova gramática $G' = \langle V', \Sigma, S, P' \rangle$ onde $V' = V \cup V_\Sigma$ e $P' = P_h \cup P_\Sigma$ onde,

$$P_h = \{\alpha \triangleright h(w_1)A_1 \cdots h(w_n)A_n h(w_{n+1}) \mid \alpha \triangleright w_1 A_1 \cdots w_n A_n w_{n+1} \in P, |n+1| \geq 2\}$$

e

$$P_\Sigma = \{A_a \triangleright a \mid A_a \in V_\Sigma\}$$

consequentemente, todo $\alpha' \triangleright \beta' \in P'$ é tal que $\beta' = X_1 \cdots X_n$ ou $\beta' = a$ com $X_k \in V'$ e $a \in \Sigma \cup \{\lambda\}$ com $1 \leq k \leq n$ e $|X_1 \cdots X_n| \geq 1$, assim por definição tem-se que G' está na forma normal pré-Chomsky. Agora para mostrar que $\mathcal{L}(G) = \mathcal{L}(G')$ basta usar um argumento similar aos utilizados nas demonstrações dos Lemas 7.1 e 7.2 e dos Teoremas 7.1, 7.4 e 7.5. ■

Teorema 7.6

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então existe uma GLC G' na forma normal de Chomsky tal que $\mathcal{L}(G) = \mathcal{L}(G')$.



Demonstração Assuma sem perda de generalidade que $G = \langle V, \Sigma, S, P \rangle$ é uma GLC assim pelo Lema 7.3 existe uma GLC na forma normal pré-Chomsky $\hat{G} = \langle \hat{V}, \Sigma, S, \hat{P} \rangle$ tal que $\mathcal{L}(G) = \mathcal{L}(\hat{G})$, e como pode ser notado na demonstração do Lema 7.3 todas as regras em \hat{P} serão das forma:

- $A \triangleright A_1 \cdots A_n$ ou
- $A \triangleright a$

com $A, A_i \in \hat{V}$ e $a \in \Sigma$ para cada $1 \leq i \leq n$. Agora defina os seguintes conjuntos disjuntos:

- $P_1 = \{A \triangleright w \in P \mid |w| \leq 2\}$
- $P_2 = P - P_1$

agora considere um conjunto de variáveis V_1 inicialmente vazio, além disso, construa um conjunto P_3 da seguinte forma, para cada $A \triangleright A_1 A_2 \cdots a_m \in P_2$ as regras:

$$\begin{aligned} A &\triangleright A_1 X_1 \\ X_1 &\triangleright A_2 X_2 \\ &\vdots \\ X_{m-2} &\triangleright A_{m-1} A_m \end{aligned}$$

são elementos de P_3 e X_i é uma nova variável adicionada ao conjunto V_1 com $i \leq m - 2$. Agora construa uma gramática $G' = \langle V', \Sigma, S, P' \rangle$ onde $V' = V_1 \cup V$ e $P' = P_1 \cup P_3$, é fácil ver que todas as regras em P' satisfazem as condições da forma normal de Chomsky, assim G' está na forma normal de Chomsky. Por fim, não é difícil verificar que para toda palavra $w \in \Sigma$ tem-se que $S \gg_G^* w$ se, e somente se, $S \gg_{G'}^* w$, consequentemente, $\mathcal{L}(G) = \mathcal{L}(G')$. ■

Para que leitores com menos facilidade com a linguagem matemática possam entender os procedimentos para encontrar a forma normal de Chomsky de uma GLC dada, agora serão convertidos os procedimentos usados nas demonstrações do Lema 7.3 e do Teorema 7.6 em dois algoritmos.

Algoritmo 4: Algoritmo para construir uma GLC na forma normal pré-Chomsky.

Entrada: Uma GLC $G = \langle V, \Sigma, S, P \rangle$

Saída: Uma GLC $G' = \langle V', \Sigma, S, P' \rangle$ na forma normal pré-Chomsky

```

1 início
2   Se existirem remova as  $\lambda$ -produções
3   Se existirem remova as regras unitárias
4   Inicialize o conjunto  $V'$  sendo igual a  $V$ 
5   Inicialize o conjunto  $P'$  como sendo vazio
6   para cada  $a \in \Sigma$  faça
7     Adicione em  $V'$  a variável  $A_a$ 
8     Adicione em  $P'$  a regra  $A_a \triangleright a$ 
9     se  $A \triangleright a \in P$  com  $A \in V$  então
10      Adicione em  $P'$  a regra  $A \triangleright a$ 
11    fim
12  fim
13  para cada  $\alpha \triangleright \beta \in P$  com  $|\beta| \geq 2$  faça
14    se  $\exists a \in \Sigma$  em  $\beta$  então
15      Gere  $\beta'$  trocando cada  $a \in \Sigma$  de  $\beta$  por  $A_a$ 
16      Adicione em  $P'$  a regra  $\alpha \triangleright \beta'$ 
17    senão
18      Adicione em  $P'$  a regra  $\alpha \triangleright \beta$ 
19    fim
20  fim
21  retorna  $G' = \langle V', \Sigma, S, P' \rangle$ 
22 fim
```

Observação: O leitor atento pode notar que a implementação do isomorfismo h descrito na demonstração do Lema 7.3 corresponde exatamente ao trecho de código entre as linhas 14 e 19 do Algoritmo 4.

Algoritmo 5: Algoritmo para construir uma GLC na forma normal de Chomsky.

Entrada: Uma GLC $G = \langle V, \Sigma, S, P \rangle$
Saída: Uma GLC $G' = \langle V', \Sigma, S, P' \rangle$ na forma normal de Chomsky

```

1 início
2   Use  $G$  como entrada do Algoritmo 4 e obtenha  $\widehat{G} = \langle \widehat{V}, \Sigma, S, \widehat{P} \rangle$ 
3   Inicialize o conjunto  $V'$  sendo igual a  $\widehat{V}$ 
4   Inicialize o conjunto  $P'$  como sendo vazio
5   para cada  $\alpha \triangleright \beta \in \widehat{P}$  com  $|\beta| \leq 2$  faça
6     | Adicione a regra  $\alpha \triangleright \beta$  no conjunto  $P'$ 
7   fim
8   para cada  $\alpha \triangleright A_1 A_2 A_3 \cdots A_n \in \widehat{P}$  com  $n \geq 3$  faça
9     | Adicione variáveis novas  $X_1, \dots, X_{n-1}$  em  $V'$ 
10    | Adicione em  $P'$  as regras  $\alpha \triangleright A_1 X_1, X_1 \triangleright A_2 X_2, \dots, X_{n-2} \triangleright A_{n-1} A_n$ 
11  fim
12  retorna  $G' = \langle V', \Sigma, S, P' \rangle$ 
13 fim
```

Exemplo 7.16 A GLC na forma normal de Chomsky obtida a partir da GLC $G = \langle \{S, A, B, C, D\}, \{0, 1\}, S, P \rangle$ com P sendo formado pelas seguintes regras:

$$\begin{aligned}
S &\triangleright 0AB \mid C1D \mid 011 \mid 100 \\
A &\triangleright 1A00 \mid 11B \mid CA0 \\
B &\triangleright 0S \mid C1S \mid \lambda \\
C &\triangleright 11C \mid B01 \mid \lambda \\
D &\triangleright 101 \mid \lambda
\end{aligned}$$

é construída aplicando primeiro o Algoritmo 4 sobre a gramática G , assim inicialmente o algoritmo remove as λ -produções gerando um conjunto de regras temporário da forma,

$$\begin{aligned}
S &\triangleright 0AB \mid C1D \mid 011 \mid 100 \mid 0A \mid C1 \mid 1D \mid 1 \\
A &\triangleright 1A00 \mid B11 \mid 0CA \mid 11 \mid 0A \\
B &\triangleright 0S \mid C1S \mid 1S \\
C &\triangleright 11C \mid B01 \mid 11 \mid 01 \\
D &\triangleright 101
\end{aligned}$$

como não produções unitárias no conjunto de regras acima o Algoritmo 4 inicializa o conjunto $V' = \{S, A, B, C, D\}$ e cria e insere em V' as variáveis A_0 e A_1 , ou seja, tem-se agora que $V' = \{S, A, B, C, D, A_0, A_1\}$ e são adicionados as regras $A_0 \triangleright 0$ e $A_1 \triangleright 1$ no conjunto de regras temporário, ou seja, tal conjunto fica da forma,

$$\begin{aligned}
S &\triangleright 0AB \mid C1D \mid 011 \mid 100 \mid 0A \mid C1 \mid 1D \mid 1 \\
A &\triangleright 1A00 \mid B11 \mid 0CA \mid 11 \mid 0A \\
B &\triangleright 0S \mid C1S \mid 1S \\
C &\triangleright 11C \mid B01 \mid 11 \mid 01 \\
D &\triangleright 101 \\
A_0 &\triangleright 0 \\
A_1 &\triangleright 1
\end{aligned}$$

agora para cada regra $\alpha \triangleright \beta$ em que $|\beta| \geq 2$ são trocados os símbolos $a \in \{0, 1\}$ de β pelas variáveis correspondentes A_a , ou seja, o conjunto de regras passa a ter a forma,

$$\begin{aligned} S &\triangleright 0AB \mid CA_1D \mid A_0A_1A_1 \mid A_1A_0A_0 \mid A_0A \mid CA_1 \mid A_1D \mid 1 \\ A &\triangleright A_1AA_0A_0 \mid BA_1A_1 \mid A_0CA \mid A_1A_1 \mid A_0A \\ B &\triangleright A_0S \mid CA_1S \mid A_1S \\ C &\triangleright A_1A_1C \mid BA_0A_1 \mid A_1A_1 \mid A_0A_1 \\ D &\triangleright A_1A_0A_1 \\ A_0 &\triangleright 0 \\ A_1 &\triangleright 1 \end{aligned}$$

neste ponto o conjunto temporário de regras já está na forma pré-Chomsky, pode-se então usar o Algoritmo 5, que irá conservar as regras da forma $\alpha \triangleright w$ com $|w| \leq 2$, e para cada regra $\alpha \triangleright A_1A_2A_3 \cdots A_n$ com $n \geq 3$ irá criar e adicionar a V' novas variáveis X_1, X_{n-1} e no conjunto de regras serão inseridas as regras $\alpha \triangleright A_1X_1, X_1 \triangleright A_2X_2, \dots, X_{n-2} \triangleright A_{n-1}A_n$, assim o conjunto de regras passa a ser da forma,

$$\begin{aligned} S &\triangleright 0X_1 \mid CX_2 \mid A_0X_3 \mid A_1X_4 \mid A_0A \mid CA_1 \mid A_1D \mid 1 \\ X_1 &\triangleright AB \\ X_2 &\triangleright A_1D \\ X_3 &\triangleright A_1A_1 \\ X_4 &\triangleright A_0A_0 \\ A &\triangleright A_1X_5 \mid BX_3 \mid A_0X_6 \mid A_1A_1 \mid A_0A \\ X_5 &\triangleright AX_4 \\ X_6 &\triangleright CA \\ B &\triangleright A_0S \mid CY_1 \mid A_1S \\ Y_1 &\triangleright A_1S \\ C &\triangleright A_1Y_2 \mid BY_3 \mid A_1A_1 \mid A_0A_1 \\ Y_2 &\triangleright A_1C \\ Y_3 &\triangleright A_0A_1 \\ D &\triangleright A_1Y_3 \\ A_0 &\triangleright 0 \\ A_1 &\triangleright 1 \end{aligned}$$

agora tem-se claramente que a gramática com tal conjunto de regras será uma GLC que está na forma normal de Chomsky.

Definição 7.19 (Forma Normal Reduzida de Greibach)

Uma GLC $G = \langle V, \Sigma, S, P \rangle$ está na forma normal reduzida de Greibach se todas as regras em P tem a forma $A \triangleright aB$ com $A \in V, a \in \Sigma$ e $B \in V^*$.



O leitor atento pode notar que as gramáticas na forma normal de Greibach possuem a propriedade de que a cada derivação um símbolo da palavra que se quer gerar é inserido na forma sentencial que é construída

pelo passo da derivação, assim nas GLC na forma normal de Greibach existe a garantia de que sempre serão necessários **exatamente** $|w|$ passos de derivação para gerar w , o que contrasta com as GLC na forma normal de Chomsky é que se sabe que para gerar w as derivações terão **no mínimo** $|w|$ passos, nesse sentido, pode-se dizer que as GLC na forma normal de Greibach são mais eficientes que as GLC na forma normal de Chomsky.

Exemplo 7.17 A GLC $G = \langle \{S, T, U\}, \{a, b\}, S, P \rangle$ onde P é definido pelas regras,

$$S \triangleright TU$$

$$T \triangleright US \mid b$$

$$U \triangleright ST \mid a$$

está na forma normal de Chomsky mas não está na forma normal de Greibach.

Exemplo 7.18 A GLC $G = \langle \{A, B, C\}, \{a, b\}, A, P \rangle$ onde P é definido pelas regras,

$$A \triangleright aC \mid a$$

$$B \triangleright b$$

$$C \triangleright aABB$$

está na forma normal de Greibach.

Teorema 7.7

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC na forma normal de Greibach, então existe uma GLC G' na forma normal de Chomsky tal que $\mathcal{L}(G) = \mathcal{L}(G')$.

Demonstração Suponha que $G = \langle V, \Sigma, S, P \rangle$ é uma GLC na forma normal de Greibach, consequentemente todas as produções são da forma $A \triangleright aB$ com $A \in V, B \in V^*$ e $a \in \Sigma$, agora construa os conjuntos,

- $V' = V \cup \{A_a \mid a \in \Sigma\}$
- $P_1 = \{A \triangleright a \in P \mid a \in \Sigma\}$
- $P_2 = P - P_1$
- $P_3 = \{A \triangleright A_a B \mid A \triangleright aB \in P_2\}$

em seguida faça $G'' = \langle V', \Sigma, S, P' \rangle$ tal que $P' = P_1 \cup P_3$, agora claramente tem-se que G'' está na forma normal pré-Chomsky e com o mesmo argumento usado na prova do Teorema 7.6 é possível construir uma GLC G' na forma normal de Chomsky tal que $\mathcal{L}(G) = \mathcal{L}(G')$. ■

Teorema 7.8

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC, então existe uma GLC G' na forma normal Greibach tal que $\mathcal{L}(G) = \mathcal{L}(G')$.

O Teorema 7.8 diz que qualquer GLC pode ser convertida para a forma normal de Greibach, neste manuscrito não será apresentado a demonstração do Teorema 7.8, caso o leitor tenha interesse em tal demonstração é sugerida a leitura de [12, 24, 27, 45].

Corolário 7.1

Se $G = \langle V, \Sigma, S, P \rangle$ é uma GLC na forma normal de Chomsky, então existe uma GLC G' na forma normal Greibach tal que $\mathcal{L}(G) = \mathcal{L}(G')$.

Demonstração Trivial. ■

A seguir é apresentado o Algoritmo que converte as GLC para a forma normal de Greibach.

Algoritmo 6: Algoritmo para construir uma GLC na forma normal de Greibach.

Entrada: Uma GLC $G = \langle V, \Sigma, S, P \rangle$ na forma normal pré-Chomsky

Saída: Uma GLC $G' = \langle V', \Sigma, S, P' \rangle$ na forma normal de Chomsky

```

1  início
2  Renomear as variáveis em  $V$  para  $A_1, \dots, A_{\#V}$  de forma que  $S$  seja renomeada como  $S_1$ 
3  Defina os três conjuntos  $P_0 = \{A_i \triangleright aB \in P \mid a \in \Sigma, B \in V^*\}$ ,  $P_1 = P - P_0$  e  $V' = V$ 
4  para  $i$  de 1 até  $\#V$  faça
5      Defina  $j = 1$ 
6      enquanto  $j < i$  faça
7          para cada  $A_i \triangleright A_j B \in P_1$  faça
8              Remova  $A_i \triangleright A_j B$  de  $P_1$ 
9              para cada  $A_j \triangleright w \in P_0 \cup P_1$  faça
10                 se  $w$  começa com algum  $a \in \Sigma$  então
11                     Adicione  $A_i \triangleright wB$  em  $P_0$ 
12                 senão
13                     Adicione  $A_i \triangleright wB$  em  $P_1$ 
14                 fim
15             fim
16         fim
17         Incremente o valor de  $j$  em 1
18     fim
19     se  $A_i$  possui regras recursivas a esquerda então
20         Adicione  $Z_i$  em  $V'$ 
21         Remova a recursão à esquerda de  $A_i$ s, onde a variável  $Z$  descrita na Definição 7.10 é
            substituída por  $Z_i$  e todas as regras iniciadas com  $a \in \Sigma$  são postas em  $P_0$  e as demais em
             $P_1$ 
22     fim
23 fim
24 para  $i$  de  $\#V - 1$  até 1 faça
25     Remova de  $P_1$  todas as regras  $A_i \triangleright A_j B$  com  $A_j \in V'$  e  $B$  sendo uma palavra sobre  $V'$ 
26     para cada  $A_j \triangleright aC \in P_0$  onde  $a \in \Sigma$  e  $C$  sendo uma palavra sobre  $V'$  faça
27         Adicione em  $P_0$  a regra  $A_i \triangleright aCB$ 
28     fim
29 fim
30 para cada  $Z_i \in V'$  com  $i \in \{1, \dots, \#V\}$  faça
31     Remova de  $P_1$  todas as regras  $Z_i \triangleright A_j B$  com  $A_j \in V'$  e  $B$  sendo uma palavra sobre  $V'$ 
32     para cada  $A_j \triangleright aC \in P_0$  onde  $a \in \Sigma$  e  $C$  sendo uma palavra sobre  $V'$  faça
33         Adicione em  $P_1$  a regra  $Z_i \triangleright aCB$ 
34     fim
35 fim
36 Defina  $P' = P_0 \cup P_1$ 
37 retorna  $G' = \langle V', \Sigma, A_1, P' \rangle$ 
38 fim

```

Exemplo 7.19 Considere a GLC do Exemplo 7.17, a mesma já está na forma normal de Chomsky e, portanto, está na forma normal pré-Chomsky, assim a mesma pode ser aplicada sem problemas ao Algoritmo 6, supondo que no passo dois o algoritmo renomeou as variáveis da seguinte forma, $S = A_1, T = A_2$ e $U = A_3$, o algoritmo segue construindo então os conjuntos,

$$P_0 = \{A_2 \triangleright b, A_3 \triangleright a\}$$

e

$$P_1 = \{A_1 \triangleright A_2A_3, A_2 \triangleright A_3A_1, A_3 \triangleright A_1A_2\}$$

e também,

$$V' = \{A_1, A_2, A_3\}$$

o algoritmo entra agora na execução do *loop* da linha 4, nas primeiras interações não são efetuadas mudanças nos conjuntos P_0 e P_1 , porém na terceira interação, isto é, quando $i = 3$ tem-se que durante o *loop* interno, iniciado na linha 6 é encontrada a regra $A_3 \triangleright A_1A_2$ em P_0 sendo excluída em seguida, logo depois a regra $A_3 \triangleright A_2A_3A_2$ é inserida em P_1 que agora tem a forma,

$$P_1 = \{A_1 \triangleright A_2A_3, A_2 \triangleright A_3A_1, A_3 \triangleright A_2A_3A_2\}$$

o valor de j aumenta para 2 e na próxima interação do *loop* (linha 6) a regra $A_3 \triangleright A_2A_3A_2$ será encontrada e excluída de P_1 em seguida será adicionado em P_0 a regra $A_3 \triangleright bA_3A_2$ e em P_1 será adicionada a regra $A_3 \triangleright A_3A_1A_3A_2$ e assim P_0 e P_1 passa a ser da forma,

$$P_0 = \{A_2 \triangleright b, A_3 \triangleright a \mid bA_3A_2\}$$

e

$$P_1 = \{A_1 \triangleright A_2A_3, A_2 \triangleright A_3A_1, A_3 \triangleright A_3A_1A_3A_2\}$$

após isso o *loop* da linha 6 é finalizando, e o algoritmo segue uma vez que A_3 possuem regra recursiva à esquerda (exatamente $A_3 \triangleright A_3A_1A_3A_2$) na linha 20 o conjunto V' é atualizado para a forma $V' = \{A_1, A_2, A_3, Z_3\}$, já os conjuntos P_0 e P_1 passam a ser da seguinte forma,

$$P_0 = \{A_2 \triangleright b, A_3 \triangleright a \mid aZ_3 \mid bA_3A_2 \mid bA_3A_2Z_3\}$$

e

$$P_1 = \{A_1 \triangleright A_2A_3, A_2 \triangleright A_3A_1, Z_3 \triangleright A_1A_3A_2 \mid A_1A_3A_2Z_3\}$$

com a execução do *loop* da linha 24 concluída o conjunto P_0 passa a conter as regras,

$$A_1 \triangleright bA_3A_2A_1A_3 \mid bA_3A_2Z_3A_1A_3 \mid bA_3 \mid aZ_3A_1A_3 \mid aA_1A_3$$

$$A_2 \triangleright b \mid bA_3A_2A_1 \mid bA_3A_2Z_3A_1 \mid aZ_3A_1 \mid aA_1$$

$$A_3 \triangleright bA_3A_2 \mid bA_3A_2Z_3 \mid aZ_3 \mid a$$

em seguida após a execução do *loop* da linha 30 concluída o conjunto P_1 irá conter as regras,

$$Z_3 \triangleright bA_3A_2A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2Z_3 \mid bA_3A_2Z_3A_1A_3A_3A_2 \mid bA_3A_2Z_3A_1A_3A_3A_2Z_3 \mid$$

$$bA_3A_3A_2 \mid bA_3A_3A_2Z_3 \mid aZ_3A_1A_3A_3A_2 \mid aZ_3A_1A_3A_3A_2Z_3 \mid aA_1A_3A_3A_2 \mid aA_1A_3A_3A_2Z_3$$

finalmente é construída a gramática $G' = \langle \{A_1, A_2, A_3, Z_3\}, \{a, b\}, A_1, P' \rangle$ onde $P' = P_0 \cup P_1$.

Pode-se como explicado em [12, 45], resumir o funcionamento do Algoritmo 6 ao receber como entrada uma GLC na forma normal pré-CHomsky nas seguintes etapas:

1. Renomear as variáveis.
2. Garantir que todas as produções seja das formas $A_i \triangleright A_jB$ ou $A_j \triangleright aB$ com $i \leq j$ e $A_i, A_j \in V, B \in V^*$ e $a \in \Sigma$.
3. Elimiar as recursões à esquerda, introduzindo com isso novas variáveis rotuladas por Z_i com $i \leq \#V$.
4. Garantir que todas as regras sejam da forma $A \triangleright aB$ com $A \in V'$ e BV'^* .

Um fato importante sobre o Algoritmo 6 é que diferentes ordens com que as variáveis são renomeadas

geram diferentes gramáticas como resultado. O exemplo a seguir ilustra essa propriedade do Algoritmo 6.

Exemplo 7.20 Considere novamente a GLC do Exemplo 7.17, e executando as etapas do Algoritmo 6, porém renomeando as variáveis agora como, $S = A_1$, $U = A_2$ e $T = A_3$ será gerado a GLC na forma normal de Greibach $G = \langle \{A_1, A_2, A_3, Z_3\}, \{a, b\}, A_1, P \rangle$ com P sendo o conjunto que contém as seguintes regras.

$$A_1 \triangleright aA_1A_2 \mid bA_2 \mid aA_1Z_3A_2 \mid bZ_3A_2$$

$$A_2 \triangleright aA_1A_2A_3 \mid bA_3A_2A_3 \mid aA_1Z_3A_2A_3 \mid bZ_3A_2A_3 \mid a$$

$$A_3 \triangleright aA_1 \mid b \mid aA_1Z_3 \mid bZ_3$$

$$Z_3 \triangleright aA_1A_2A_3A_3A_1 \mid bA_3A_2A_3A_3A_1 \mid aA_1Z_3A_2A_3A_3A_1 \mid bZ_3A_2A_3A_3A_1 \mid aA_3A_1 \\ aA_1A_2A_3A_3A_1Z_3 \mid bA_3A_2A_3A_3A_1Z_3 \mid aA_1Z_3A_2A_3A_3A_1Z_3 \mid bZ_3A_2A_3A_3A_1Z_3 \mid aA_3A_1Z_3$$

7.3 Sobre Algoritmos de Pertinência em GLC

Escrever depois...

7.4 Sobre GLC e Linguagens de Programação

Escrever depois...

7.5 Autômatos de Pilha

Nas Seções 6.1-6.3 do Capítulo 6 foram apresentados uma série de três classes de máquinas (os autômatos) que era capazes de reconhecer (ou computar) linguagens regulares. Nesta seção e na próxima seção este manuscrito irá continuar com este caminho de apresentar as máquinas de computação, ou seja, serão apresentadas modelos de máquina (autômatos) para reconhecer (ou computar) linguagens livres do contexto.

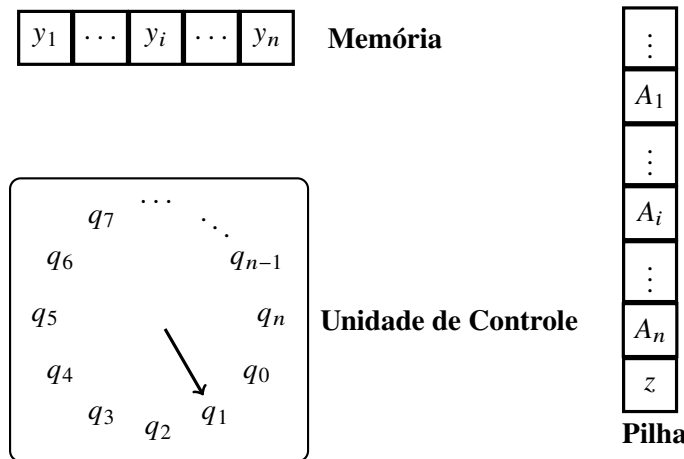


Figura 7.4: Conceito informal de autômato de pilha.

Um autômato de pilha pode ser compreendido como sendo um computador com uma estrutura formado por três partes (a) uma memória somente de leitura, (b) uma unidade de controle e (c) uma memória de escrita e leitura na forma de uma pilha. A figura 7.4 ilustra essa visão conceitual da estrutura dos autômatos de pilha.

Os dois primeiros componentes dos autômatos de pilha são similares aquele que o leitor já encontrou no Capítulo 6 quando esteve em contato com os AFD, AFN e λ -AFN. O terceiro componente (a pilha) é uma

memória⁴ cujo acesso obedece os critérios de uma estrutura de dados d tipo pilha [60], em tal memória será sempre possível ler o topo da pilha e remover (fazer *pop*) ou escrever (*push*) sob o topo da pilha.

Para simplificar o entendimento inicial do leitor será primeiro apresentado um modelo determinístico do Autômatos de pilha, o modelo aqui apresentado é o mesmo usado em [12], vale salientar que este modelo é equivalente em poder computacional aos modelos usuais usados em [31, 33], ou seja, ele reconhece a mesma classe de linguagens. O modelo usual encontrado na literatura será detalhado e usado mais adiante neste manuscrito.

Definição 7.20 (Autômato de Pilha Determinístico)

Um Autômato de Pilha Determinístico, ou simplesmente APD, é uma estrutura $A = \langle Q, \Sigma, \Gamma, \delta, q_0, z, F \rangle$ onde, Q é um conjunto finito e não vazio de estados, Σ é o alfabeto de entrada, Γ é o alfabeto da pilha, $\delta : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$ é uma função total de transição entre os estados do autômato, $q_0 \in Q$ é o estado inicial do APD, $z \in \Gamma$ é um símbolo especial chamado de símbolo de fim da pilha^a e $F \subseteq Q$ é um conjunto de estados finais

^aPara o leitor conhecedor dos fundamentos de estrutura de dados e programação, o símbolo da pilha vazia pode ser visto como sendo o ponteiro que aponta para *null* indicado o fim da pilha.



Observação: Em obras como [12, 31, 33, 45] o símbolo z costuma ser chamado de símbolo da pilha vazia.

Exemplo 7.21 A Estrutura $A = \langle \{q_0, q_1\}, \{a, b\}, \{A, z\}, \delta, q_0, z, \{q_1\} \rangle$ com δ definido a seguir é um APD.

$$\begin{aligned} \delta(q_0, a, z) &= (q_0, Az) \\ \delta(q_0, a, A) &= (q_0, AA) \\ \delta(q_0, b, z) &= (q_0, z) \\ \delta(q_0, b, A) &= (q_1, \lambda) \\ \delta(q_1, a, z) &= (q_0, z) \\ \delta(q_1, a, A) &= (q_0, \lambda) \\ \delta(q_1, b, z) &= (q_0, z) \\ \delta(q_1, b, A) &= (q_1, \lambda) \end{aligned}$$

Pode-se interpretar que $\delta(q_i, a, A) = (q_j, A_1 A_2 \cdots A_n)$ como sendo uma instrução com o seguinte significado, estando o autômato (ou a máquina) no estado q_i e se o cabeçote da memória estiver sobre a cédula que contém a e, além disso, estando o símbolo A no top da pilha, então o autômato muda para estado q_j e a palavra $A_1 A_2 \cdots A_n$ é empilhada⁵ no top da pilha. Sobre as transições, elas podem ser categorizadas em 3 classes:

- Transições de consulta, são aquelas que o símbolo exigido no topo da pilha é o mesmo que será escrito, ou seja, a transição tem a forma $\delta(q_i, a, A) = (q_i, A)$.
- Transições de inserção (ou *push*), são aquelas que aumentam o tamanho da pilha, ou seja, essas transições expandem o tamanho da pilha, o leitor reconhecerá essas transições pela forma $\delta(q_i, a, A) = (q_j, A_1 \cdots A_n)$ com $n \geq 2$.
- Transições de remoção (ou *pop*), são aquelas que diminuem o tamanho da pilha, ou seja, essas transições

⁴Assim com a memória só de escrita, a pilha tem a natureza de ser dividida em cédulas, em que cada cédula pode armazenar o único símbolos por vez.

⁵Como dito é [12], quando uma palavra $A_1 A_2 \cdots A_n$ é empilhada no top da pilha, os símbolos da pilha sofrem *push* de forma reversar, assim é A_n substitui o atual topo da pilha, depois A_{n-1} é inserido na cédula sobre a cédula que guarda A_n , e isso se repete até que A_1 seja colocada na pilha, por fim o topo da pilha passa a ser A_1 .

reduzem o tamanho da pilha, tais transições tem a forma $\delta(q_i, a, A) = (q_j, \lambda)$.

Agora que já foram introduzidos os conceitos básicos a cerca dos autômatos de pilha em sua versão mais simples, isto é, na versão determinística, este manuscrito irá prosseguir apresentando a versão não-determinística.

Definição 7.21 (Autômato de Pilha Não-determinístico)

Um Autômato de Pilha Determinístico, ou simplesmente APN, é uma estrutura $A = \langle Q, \Sigma, \Gamma, \delta, q_0, z, F \rangle$ onde, $Q, \Sigma, \Gamma, q_0, z, F$ são da mesma forma que na Definição 7.20 e δ é uma função parcial com a seguinte assinatura $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow \wp^{fin}(Q \times \Gamma^*)$.

Observação: Como dito em [31] na notação $\wp^{fin}(Q \times \Gamma^*)$ denota uma família finita de subconjuntos de $Q \times \Gamma^*$.

Exemplo 7.22 A Estrutura $A = \langle \{q_0, q_1, q_2\}, \{a, b\}, \{A, z\}, \delta, q_0, z, \{q_2\} \rangle$ com δ definido como sendo,

$$\begin{aligned}\delta(q_0, a, z) &= \{(q_0, AAz)\} \\ \delta(q_0, a, A) &= \{(q_0, AA)\} \\ \delta(q_0, b, A) &= \{(q_1, \lambda)\} \\ \delta(q_1, b, A) &= \{(q_1, \lambda)\} \\ \delta(q_1, c, A) &= \{(q_2, \lambda)\} \\ \delta(q_2, c, A) &= \{(q_2, \lambda)\}\end{aligned}$$

é um APN.

Um ponto a se destacar neste momento é que a versão determinística dos autômatos de pilha apresentados em obras como [31, 33, 57] consiste na verdade de um APN que para todo $q \in Q, a \in \Sigma \cup \{\lambda\}$ e $A \in \Sigma$ obedece as seguintes restrições:

- (1) $\#\delta(q, a, A) \leq 1$ e
- (2) Se $\delta(q, \lambda, A) \neq \emptyset$, então $\delta(q, a, A) = \emptyset$.

O leitor atento pode notar que o APN no Exemplo 7.22 é na verdade um APD.

Com respeito a representação visual, um APN (ou APD) pode ser representado usando grafos de transição, como para o caso dos autômatos do Capítulo 6 os estados são as arestas que são desenhada com círculos usando a seguinte regra se $q \in F$ então é desenhado círculos duplo se não é um círculo simples, já no caso de $(q_j, A_1 \cdots A_n) \in \delta(q_i, a, A_0)$ então é desenhada um vértice ligando as arestas q_i e q_j , tal que este vértice é rotulado por $a, A_0 \mid A_1 \cdots A_n$, com $a \in \Sigma \cup \{\lambda\}$ e $A_i \in \Gamma^*$ para $0 \leq i \leq n - 1$.

Exemplo 7.23 O APN do Exemplo 7.22 pode ser representado pelo seguinte grafo de transição

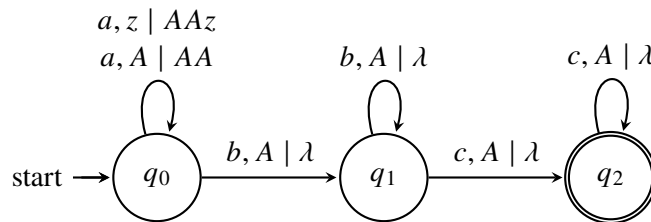


Figura 7.5: Grafo de transição do APN do Exemplo 7.22.

A qualquer momento no tempo pode-se com dito em [31] descrever a atual configuração de um APN (ou APD) através do conhecimento de três informações: O estado atual do APN (ou APD), o que resta para ser processado na memória e conteúdo da pilha.

Definição 7.22 (Descrição Instantânea)

Dado um APN (APD) $A = \langle Q, \Sigma, \Gamma, \delta, q_0, z, F \rangle$ uma descrição instantânea (d.i.) de A , é uma tripla (q, w, W) com $q \in Q, w \in \Sigma^*$ e $W \in \Gamma^*$. O conjunto de todas as descrições instantâneas sobre A é denotado por DI_A .

Observação: Sempre que não causar confusão a escrita DI_A pode ser modificada para DI .

Sobre o conjunto de todas as descrições instantâneas é possível definir uma relação de movimento (ou mudança), tal relação na verdade descreve a execução de um passo de computação em um autômato de pilha, independente de ser determinístico ou não.

Definição 7.23 (Relação de Movimento)

Dado um APN (APD) $A = \langle Q, \Sigma, \Gamma, \delta, q_0, z, F \rangle$ a relação de movimento sobre DI , denotada por \rightarrow , é dada por,

$$(q_i, aw, A\beta z) \rightarrow (q_j, w, \alpha\beta z) \iff (q_j, \alpha) \in \delta(q_i, a, A) \quad (7.1)$$

Por sua vez, o fecho transitivo e reflexivo de \rightarrow é denotado por \rightarrow^* .

Exemplo 7.24 Considere o APN do Exemplo 7.22 tem-se que,

$$(q_0, aaaabbcc, z) \rightarrow (q_0, aaabbcc, AAz)$$

pois $(q_0, AAz) \in (q_0, a, z)$. Além disso, é fácil mostrar que

$$(q_0, aaaabbcc, z) \rightarrow^* (q_2, \lambda, z)$$

Após apresentar a relação de movimento é possível seguir em frente e formalizar o conceito de palavra aceita por autômatos de pilha, como apresentado em [31] existe no mínimo duas forma com que um autômato de pilha aceita uma palavra de entrada, a primeira seria o autômato simplesmente após lê toda a palavra de entrada atingir um estado final e a segunda opção que é aquela em que o autômato deve lê toda a palavra de entrada, atingir um estado final e ter a pilha vazia⁶.

Definição 7.24 (Linguagem Aceita por Estado final)

Dado um APN (APD) $A = \langle Q, \Sigma, \Gamma, \delta, q_0, z, F \rangle$ a linguagem de A aceita por estado final corresponde ao seguinte conjunto,

$$\mathcal{L}_F(A) = \{w \in \Sigma^* \mid (q_0, w, z) \rightarrow^* (q_f, \lambda, u), q_f \in F, u \in \Gamma^*\} \quad (7.2)$$

Definição 7.25 (Linguagem Aceita por Pilha vazia)

Dado um APN (APD) $A = \langle Q, \Sigma, \Gamma, \delta, q_0, z, F \rangle$ a linguagem de A aceita por pilha vazia corresponde ao seguinte conjunto,

$$\mathcal{L}_z(A) = \{w \in \Sigma^* \mid (q_0, w, z) \rightarrow^* (q_f, \lambda, z), q_f \in F\} \quad (7.3)$$

Como mostrado em [33] uma linguagem L será aceita por um APN A por pilha vazia se, e somente se, existir outro APN A' que aceita L por estado final.

Observação: Para esse questionário sempre que $w \in \Sigma^*$ e $c \in \Sigma$ a notação $|w|_c$ irá representar o número de c 's que existem na palavra w .

⁶Na visão (praticamente dogmática) do autor deste manuscrito, ele acredita que a segunda opção é mais compatível com a natureza dos APN e APD como sendo computadores munidos de ferramenta de controle por contagem.

Questionário do Capítulo

1. Prove que a linguagem $\{w \in \{0, 1\}^* \mid |w|_0 = |w|_1 \text{ e } |u|_0 \geq |u|_1, u \in PRE(w)\}$ é gerada pela GLC com o seguinte conjunto de produções,

$$S \rhd 0S1 \mid SS \mid \lambda \quad (7.4)$$

2. Usando a GLC do exercício anterior desenhe as árvores de derivação para as palavras *abaabb*, *aabbaabb* e *abababab*.

3. Exiba uma GLC para cada linguagem a seguir.

- (a). $\{a^n b^m \mid n \neq m - 1\}$
- (b). $\{1^n 0^m \mid n \neq 2m\}$
- (c). $\{0^n 1^m \mid n \leq m + 4\}$
- (d). $\{a^n b^m \mid 2m \leq n + 1\}$
- (e). $\{a^n b^m c^k \mid m = n \text{ ou } m \leq k\}$
- (f). $\{a^n b^m c^k \mid k = n \text{ ou } m \neq n\}$
- (g). $\{a^{2n} b^n c^n \mid n \geq 2\}$
- (h). $\{w \in \{0, 1, 2\}^* \mid |w|_1 = 2k + 1, k \in \mathbb{N}\}$
- (i). $\{0^i b^j a^k \mid i, k \in \mathbb{N}, j = 2k + i\}$
- (j). $\{0^i b^j a^k \mid i, k \in \mathbb{N}, 2i = j + k\}$
- (k). $\{0^i b^j a^k \mid j, k \in \mathbb{N}, i = 2k + j + 3\}$
- (l). $\{a^i b^j c^k \mid j, k \in \mathbb{N}, j = 2(i + k) + 1\}$
- (m). $\{wu^n w^r \mid u \in \{0, 1\}^+, w \in \{a, b\}^*, n \in \mathbb{N}\}$

4. Considere a linguagem $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ prove para todo $k \in \mathbb{N}$ que a linguagem L^k é livre do contexto.

5. Prove que $00110110 \notin \mathcal{L}(G)$ onde G é a GLC que tem o seguinte conjunto de regras e S sendo o símbolo inicial.

$$\begin{aligned} S &\rhd 00Y \\ X &\rhd 1Y1 \mid \lambda \\ Y &\rhd X0 \end{aligned}$$

6. Remova a recursão à esquerda da GLC com o seguinte conjunto de regras com $\Sigma = \{0, 1\}$ e S sendo o símbolo inicial.

$$S \rhd 010S \mid SS \mid SS01 \mid 01$$

7. Remova a recursão à esquerda da GLC com o seguinte conjunto de regras com $\Sigma = \{0, 1\}$ e S sendo o símbolo inicial.

$$\begin{aligned} S &\rhd S1 \mid AS \mid \lambda \\ A &\rhd A0 \mid 0 \end{aligned}$$

8. Remova a recursão à esquerda da GLC com o seguinte conjunto de regras com $\Sigma = \{0, 1, 2\}$ e S sendo o símbolo inicial.

$$\begin{aligned} S &\rhd 2Z \mid S10S \\ Z &\rhd 2Z1 \mid \lambda \end{aligned}$$

9. Construa uma GLC sem recursão à esquerda que gerem a linguagem $\{w \in \{0, 1\}^* \mid |w|_1 > |w|_0\}$.

10. Elimine as regras inúteis da GLC com o conjunto de regras a seguir e sendo $\Sigma = \{a, b, c, d\}$ e S sendo o símbolo inicial.

$$S \rightarrow a \mid aB \mid B \mid C$$

$$A \rightarrow aB \mid \lambda$$

$$B \rightarrow Aa$$

$$C \rightarrow cCD$$

$$D \rightarrow dd \mid ddD$$

11. Elimine as λ -produções da GLC com o conjunto de regras a seguir e sendo $\Sigma = \{0, 1\}$ e S sendo o símbolo inicial.

$$S \rightarrow A0B \mid 0C0B$$

$$A \rightarrow 0B0 \mid \lambda \mid B$$

$$B \rightarrow 11A \mid \lambda$$

$$C \rightarrow 111C \mid AB$$

12. Considerando $\Sigma = \{0, 1\}$ e S remova todas as λ -produções, regras unitárias e recursão à esquerda das GLC a seguir.

- (a). Com S sendo a variável inicial e P é formado por,

$$S \rightarrow S0 \mid 0AB$$

$$A \rightarrow B \mid aa \mid \lambda$$

$$B \rightarrow A1B \mid \lambda \mid C$$

$$C \rightarrow 0CC0 \mid ABC$$

- (b). Com A sendo a variável inicial e P é formado por,

$$S \rightarrow 0S0 \mid \lambda$$

$$A \rightarrow SASA \mid B$$

$$B \rightarrow 1B \mid C \mid D1$$

$$C \rightarrow 0C1 \mid 01$$

$$D \rightarrow D0 \mid 1D$$

- (c). Com C sendo a variável inicial e P é formado por,

$$S \rightarrow 0A \mid 0B \mid S0A$$

$$A \rightarrow B \mid 00 \mid \lambda$$

$$B \rightarrow 1B \mid \lambda \mid C$$

$$C \rightarrow 0C0 \mid ABC$$

- (d). Com B sendo a variável inicial e P é formado por,

$$S \rightarrow S10A \mid A0B \mid S0A11$$

$$A \rightarrow AB \mid A00 \mid 00 \mid \lambda$$

$$B \rightarrow 1B \mid \lambda \mid BC \mid BBS \mid BB0S$$

$$C \rightarrow B \mid CBA \mid CA \mid A \mid C \mid 1 \mid \lambda$$

13. Converta para a forma normal pré-Chomsky todas as GLC dos exercício de 5 até 8.
14. Converta para a forma normal pré-Chomsky as GLC dos exercício de 10 e 11.
15. Converta para a forma normal Chomsky as GLC dos exercício de 13 e 14.
16. Converta para a forma normal Chomsky todas as GLC do exercício 12, 13, 14.
17. Converta para a forma normal Greibach todas as GLC do exercício 12, 13, 14.
18. Esboce um APN ou APD para cada linguagem do exercício 3.

VI

λ -CÁLCULO

Alice perguntou, “-Gato Cheshire... pode me dizer qual o caminho que eu devo tomar?”

“-Isso depende muito do lugar para onde você quer ir”, disse o Gato.

“-Eu não sei para onde ir!”, respondeu Alice.

“-Se você não sabe para onde ir, qualquer caminho serve”. Falou o gato.

LEWIS CARROLL, ALICE NO PAÍS DAS MARAVILHAS.

VII

COMPLEXIDADE

Alice perguntou, “-Gato Cheshire... pode me dizer qual o caminho que eu devo tomar?”

“-Isso depende muito do lugar para onde você quer ir”, disse o Gato.

“-Eu não sei para onde ir!”, respondeu Alice.

“-Se você não sabe para onde ir, qualquer caminho serve”. Falou o gato.

LEWIS CARROLL, ALICE NO PAÍS DAS MARAVILHAS.

Bibliografia

- [1] Jair Minoro Abe. *Introdução à Lógica para a Ciência da Computação*. Arte & Ciência, 2002.
- [2] Jair Minoro Abe e Nelson Papavero. *Teoria Intuitiva dos Conjuntos*. MAKRON Books, 1991.
- [3] Alfred V AHO et al. *Compiladores: Princípios, Técnicas e ferramentas*. 2ª Edição. Editora Pearson, 2007.
- [4] Edgard de Alencar Filho. *Iniciação à Lógica Matemática*. NBL Editora, 2002.
- [5] Whitehead AN e B Russell. *Principia Mathematica*. 1910.
- [6] Jeremy Avigad. «Handbook of proof theory». Em: *Studies in Logic and the Foundations of Mathematics*, ch. Citeseer. 1998.
- [7] Mauricio Ayala-Rincón e F L Cavalcanti de Moura. *Fundamentos da Programação Lógica e Funcional – O princípio de Resolução e a Teoria de Reescrita*. Editora UnB, 2014.
- [8] Jorge Muniz Barreto et al. «Fundamentos de Matemática Aplicada a Informática». Acessado em 06/06/2021 na página <http://www.inf.ufsc.br/~mauro.roisenberg/ine5381/leituras/apostila.pdf>. 1998.
- [9] Benjamín Bedregal e Benedito Melo Acióly. «Introdução à Lógica Clássica para a Ciência da Computação». Notas de aula. 2007.
- [10] Benjamín Bedregal. « λ -ALN: Autômatos Lineares Não-determinísticos com λ -Transições». Em: *TEMA - Tendências em Matemática Aplicada e Computacional* 12.3 (2011), pp. 171–182.
- [11] Benjamín Bedregal. «Nondeterministic Linear Automata and a Class of Deterministic Linear Languages». Em: *Preliminary Proceedings LSFA* (2015), pp. 183–196.
- [12] Benjamín Bedregal, B M Acióly e A Lyra. *Introdução à Teoria da Computação: Linguagens Formais, Autômatos e Computabilidade*. Natal: Editora UnP, 2010.
- [13] Yves Bertot e Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer Science & Business Media, 2013.
- [14] Samuel R Buss. *Handbook of proof theory*. Elsevier, 1998.
- [15] Georg Cantor. «Beiträge zur Begründung der Transfiniten Mengenlehre». Em: *Mathematische Annalen* 46.4 (1895), pp. 481–512.
- [16] José Carmo, Paula Gouveia e Francisco Miguel Dionísio. *Elementos de Matemática Discreta*. College Publications, 2013.
- [17] John Carroll e Darrell Long. *Theory of finite automata: with an introduction to formal languages*. 2ª Edition. New Jersey: Prentice Hall Upper Saddle River, NJ, 1989.
- [18] Noam Chomsky. «On certain formal properties of grammars». Em: *Information and control* 2.2 (1959), pp. 137–167.
- [19] Noam Chomsky. «Three models for the description of language». Em: *IRE Transactions on information theory* 2.3 (1956), pp. 113–124.
- [20] Irving M Copi. *Introdução à Lógica*. Mestre Jou, 1981.

- [21] Valdigeis S Costa. «Autômatos Fuzzy Hesitantes Típicos: Teoria e Aplicações». Tese de doutoramento. Natal, RN: Programa de Pós-graduação em Sistemas e Computação, Universidade Federal do Rio Grande do Norte, UFRN, 2020.
- [22] Valdigeis S Costa. «Linguagens Lineares Fuzzy». Tese de mestrado. Natal, RN: Programa de Pós-graduação em Sistemas e Computação, Universidade Federal do Rio Grande do Norte, UFRN, 2016.
- [23] Colin De la Higuera. *Grammatical inference: Learning Automata and Grammars*. London: Cambridge University Press, 2010.
- [24] Grzegorz Ehrenfeucht Andrzej Rozenberg. «An Easy Proof of Greibach Normal FORM». Em: *Information and Control* 63 (1984), pp. 190–199.
- [25] Antonio Diego S Farias et al. «A Residuated Function in a Class of Mealy Type \mathcal{L} -Valued Finite Automaton». Em: *Fuzzy Information Processing Society (NAFIPS), 2016 Annual Conference of the North American*. IEEE. El Paso, TX, USA, 2016, pp. 1–6.
- [26] Kurt Gödel. «Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I». Em: *Monatshefte für mathematik und physik* 38.1 (1931), pp. 173–198.
- [27] Sheila A Greibach. «A new normal-form theorem for context-free phrase structure grammars». Em: *Journal of the ACM (JACM)* 12.1 (1965), pp. 42–52.
- [28] Paul R Halmos. *Teoria ingênua dos conjuntos*. Editora Ciência Moderna, 2001.
- [29] Joseph Halpern, Zohar Manna e Ben Moszkowski. «A hardware Semantics Based on Temporal Intervals». Em: *International Colloquium on Automata, Languages, and Programming*. Springer. 1983, pp. 278–291.
- [30] David Harel et al. «First-order Dynamic Logic». Em: *Lecture Notes Computer Sciences* 9 (1979), p. 133.
- [31] Michael A Harrison. *Introduction to formal language theory*. Addison-Wesley Longman Publishing Co., Inc., 1978.
- [32] Wilfrid Hodges et al. *A Shorter Model Theory*. Cambridge university press, 1997.
- [33] John E Hopcroft, Rajeev Motwani e Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. 3ª. USA: Pearson Education India, 2008.
- [34] Stephen Cole Kleene. *Representation of events in nerve nets and finite automata*. Rel. téc. Rand Project Air Force Santa Monica CA, 1951.
- [35] Peter Linz. *An Introduction to Formal Languages and Automata*. USA: Jones & Bartlett Learning, 2006.
- [36] Seymour Lipschutz. *Teoria dos Conjuntos*. McGraw-Hill do Brasil, 1978.
- [37] Seymour Lipschutz e Marc Lipson. *Matemática Discreta*. Coleção Schaum. Bookman Editora, 2013.
- [38] Carlos Alberto Lungarzo. «La Consistencia de la Lógica Intuicionista». Em: *Tarea* 3 (1972), pp. 119–132.
- [39] P. D. Magnus et al. *forall x: Calgary. An Introduction to Formal Logic*. Fall 2020, 2020.
- [40] Zohar Manna e Amir Pnueli. «The Modal Logic of Programs». Em: *International Colloquium on Automata, Languages, and Programming*. Springer. 1979, pp. 385–409.
- [41] John C. Martin. *Introduction to Languages and The Theory of Computation*. 4ª Edição. McGraw Hill, 2003.

- [42] João Paiva Martins. *Lógica e Raciocínio*. College Publications, 2014.
- [43] Warren S McCulloch e Walter Pitts. «A logical calculus of the ideas immanent in nervous activity». Em: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133.
- [44] George H Mealy. «A method for synthesizing sequential circuits». Em: *The Bell System Technical Journal* 34.5 (1955), pp. 1045–1079.
- [45] Paulo B. Menezes. *Linguagens Formais e Autômatos*. Sagra-Dcluzzato, 1998.
- [46] Paulo B. Menezes. *Matemática Discreta para Computação e Informática*. Vol. 2. Bookman, 2010.
- [47] Edward F Moore. «Gedanken-experiments on sequential machines». Em: *Automata Studies* 34 (1956), pp. 129–153.
- [48] John Myhill. «Creative sets». Em: *Journal of Symbolic Logic* 22.1 (1957).
- [49] Anil Nerode. «Linear automaton transformations». Em: *Proceedings of the American Mathematical Society* 9.4 (1958), pp. 541–544.
- [50] Roger Pressman e Bruce Maxim. *Engenharia de Software*. 8ª Edição. McGraw Hill Brasil, 2016.
- [51] Michael O Rabin. «Probabilistic automata». Em: *Information and Control* 6.3 (1963), pp. 230–245.
- [52] Michael O Rabin e Dana Scott. «Finite automata and their decision problems». Em: *IBM Journal of Research and Development* 3.2 (1959), pp. 114–125.
- [53] Henry Gordon Rice. «Classes of recursively enumerable sets and their decision problems». Em: *Transactions of the American Mathematical Society* 74.2 (1953), pp. 358–366.
- [54] Masahiko Sato et al. «Calculi of Meta-variables». Em: *International Workshop on Computer Science Logic*. Springer. 2003, pp. 484–497.
- [55] Dana S Scott e Christopher Strachey. *Toward a mathematical semantics for computer languages*. Vol. 1. Oxford University Computing Laboratory, Programming Research Group Oxford, 1971.
- [56] João Inácio da Silva Filho. «Lógica Paraconsistente e Probabilidade Pragmática no Tratamento de Incertezas». Em: *Revista Seleção Documental* 9 (2008), pp. 16–27.
- [57] Michael Sipser. *INTRODUÇÃO À TEORIA DA COMPUTAÇÃO-2® EDIÇÃO NORTE-AMERICANA*. Cengage Learning Edições Ltda., 2010.
- [58] Ian Sommerville. *Software Engineering*. 9ª Edição. Pearson, 2011.
- [59] João Nunes de Souza. *Lógica para Ciência da Computação e Áreas Afins*. Elsevier Brasil, 2008.
- [60] Jayme Luiz Szwarcfiter e Lilian Markenzon. *Estruturas de Dados e seus Algoritmos*. Vol. 2. Livros Tecnicos e Científicos, 1994.
- [61] Alfred Tarski. *Logic, Semantics, Metamathematics: Papers From 1923 To 1938*. Hackett Publishing, 1983.
- [62] Alan Mathison Turing. «On Computable Numbers, with an Application to the Entscheidungsproblem». Em: *Proceedings of the London mathematical society* 2.1 (1937), pp. 230–265.
- [63] Daniel J Velleman. *How to prove it: A structured approach*. Cambridge University Press, 2019.