Capítulo 4

Propriedades das Linguagens Regulares

Estamos no momento de colocar a seguinte questão: quão geral são as linguagens regulares? Seria toda linguagem formal regular? Talvez qualquer conjunto que possamos especificar seja aceito por algum autômato finito, embora complexo. Como veremos essa conjectura é falsa. Para entender essa resposta, devemos nos aprofundar na natureza das linguagens regulares e ver que propriedades a família como um todo tem.

A primeira questão que levantaremos é o que acontece quando operamos com linguagens regulares. As operações que consideraremos são operações tais como concatenação, união, etc. É a linguagem resultante ainda regular? Nos referiremos a isso como uma questão de *fecho*. Propriedades de fecho, embora, principalmente, de interesse teórico, permiti-nos-á discriminar muitas linguagens regulares que encontraremos.

A segunda questão que levantaremos trata de nossa habilidade para decidir sobre certas propriedades. Por exemplo, podemos fazer um algoritmo que decida se uma linguagem regular arbitrária é finita ou não? Como veremos, tais questões são facilmente respondidas para a classe das linguagens regulares, mas não para outras classes de linguagens.

Finalmente, consideremos a importante questão: como podemos dizer se uma dada linguagem é regular ou não? Se a linguagem é de fato regular, podemos sempre mostrar isso exibindo um AFD que a reconhece, ou uma expressão regular que a denota ou uma gramática regular que a gera. Mas se não for o caso, precisaríamos de outra abordagem, pois o fato de não termos conseguido encontrar um AFD que reconheça a linguagem não significa necessariamente que tal autômato não exista. Uma maneira de mostrar que uma linguagem não é regular é estudar as propriedades gerais das linguagens regulares.

4.1 Propriedades de Fecho de Linguagens Regulares

Considere a seguinte questão: dadas duas linguagens regulares arbitrárias $\mathcal{L}_1$ e $\mathcal{L}_2$, é sua intersecção também regular? Em exemplos específicos a resposta pode ser óbvia, mas aqui pretendemos atacar o problema em geral. Formularemos questões análogas para as demais operações.

Teorema 4.1.1 *Se $\mathcal{L}_1$ e $\mathcal{L}_2$ são linguagens regulares, então $\mathcal{L}_1 \cup \mathcal{L}_2$, $\mathcal{L}_1\mathcal{L}_2$, $\mathcal{L}_1^*$, $\overline{\mathcal{L}_1}$, $\mathcal{L}_1 \cap \mathcal{L}_2$ e $\mathcal{L}_1 - \mathcal{L}_2$ também são. Dizemos com isso que a família das linguagens regulares é fechada sob união, concatenação, fecho estrela, complemento e intersecção.*

DEMONSTRAÇÃO: Se $\mathcal{L}_1$ e $\mathcal{L}_2$ são linguagens regulares, então existem expressões regulares r_1 e r_2 tais que $L(r_1) = \mathcal{L}_1$ e $L(r_2) = \mathcal{L}_2$.

União e fecho estrela: Da definição 3.2.1, temos

$$\mathcal{L}_1 \cup \mathcal{L}_2 = L(r_1) \cup L(r_2) = L(r_1 + r_2)$$

$$\mathcal{L}_1\mathcal{L}_2 = L(r_1)L(r_2) = L(r_1r_2)$$

$$\mathcal{L}_1^* = (L(r_1))^* = L(r_1^*)$$

Portanto, o fecho sob união, concatenação e fecho-estrela é imediato.

Complemento: Seja $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ um AFD que aceita $\mathcal{L}_1$. Então, trivialmente, o AFD

$$\widehat{M} = \langle Q, \Sigma, \delta, q_0, Q - F\rangle,$$

aceita $\overline{\mathcal{L}_1}$. Observe que na definição de um AFD, assumimos que δ^* era uma função total, ou seja $\delta^*(q_0, w)$ está definida para todo $w \in \Sigma^*$. Conseqüentemente, ou $\delta^*(q_0, w)$ é um estado final, e nesse caso $w \in \mathcal{L}_1$, ou não ($\delta^*(q_0, w) \in Q - F$) e nesse caso $w \in \overline{\mathcal{L}_1}$.

Intersecção: Sejam $M_1 = \langle Q, \Sigma_1, \delta_1, q_0, F_1\rangle$ e $M_2 = \langle P, \Sigma_2, \delta_2, p_0, F_2\rangle$ AFD's que reconhecem $\mathcal{L}_1$ e $\mathcal{L}_2$, ou seja $\mathcal{L}_1 = L(M_1)$ e $\mathcal{L}_2 = L(M_2)$. Vamos construir a partir de M_1 e M_2 o autômato

$$\widehat{M} = \langle \widehat{Q}, \Sigma_1 \cap \Sigma_2, \widehat{\delta}, (q_0, p_0), \widehat{F}\rangle,$$

cujo conjunto de estados, $\widehat{Q} = Q \times P$, consiste de pares (q_i, p_j), e cuja função de transição $\widehat{\delta}$ é tal que $\widehat{M}$ está no estado (q_i, p_j), se M_1 está no estado q_i e M_2 está no estado p_j. Isto é conseguido tomando

$$\widehat{\delta}((q_i, p_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a)).$$

$\widehat{F}$ é definido como o conjunto de todos os (q_i, p_j) tal que $q_i \in F_1$ e $p_j \in F_2$, isto é, $\widehat{F} = F_1 \times F_2$.

É fácil ver que $w \in \mathcal{L}_1 \cap \mathcal{L}_2$ se, e somente se, $w \in L(\widehat{M})$. Assim, $L(\widehat{M}) = L(M_1) \cap L(M_2)$. Conseqüentemente, $\mathcal{L}_1 \cap \mathcal{L}_2$ é regular.

Diferença: A família das linguagens regulares é fechada com respeito à diferença, se quando $\mathcal{L}_1$ e $\mathcal{L}_2$ são linguagens regulares, então $\mathcal{L}_1 - \mathcal{L}_2$ também é regular. Mas

$$\mathcal{L}_1 - \mathcal{L}_2 = \mathcal{L}_1 \cap \overline{\mathcal{L}_2}.$$

Como já mostramos aqui que as linguagens regulares são fechadas sobre a intersecção e complemento, podemos concluir que $\mathcal{L}_1 - \mathcal{L}_2$ é uma linguagem regular. ■

Exemplo 4.1.2 *Sejam M_1 e M_2 os* AFD*'s descritos nas figuras 4.1 e 4.2, respectivamente. A intersecção das linguagens $L(M_1)$ com $L(M_2)$, é reconhecida pelo autômato do exercício 2.2.6, o qual, se substituirmos PP por (q_0, q_0), PI por (q_0, q_1), IP por (q_1, q_0) e II por (q_1, q_1), é exatamente o mesmo autômato que o construído usando o mecanismo descrito no teorema anterior, para a intersecção.*

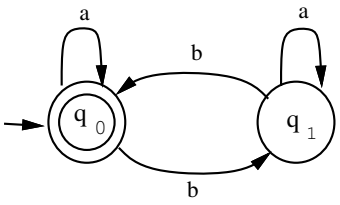


Figura 4.1: AFD M_1

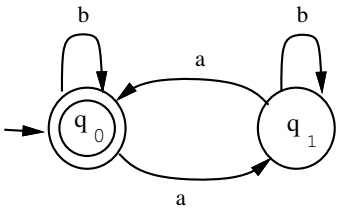
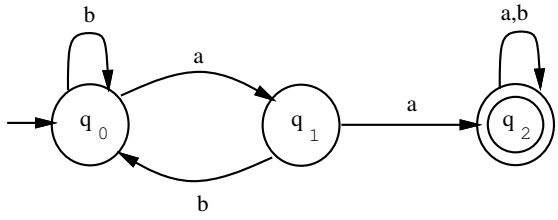


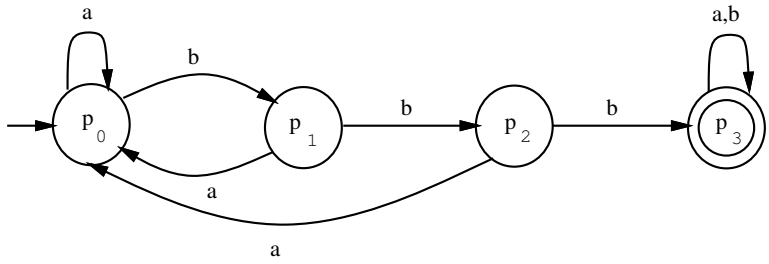
Figura 4.2: AFD M_2

Exemplo 4.1.3 *Sejam as linguagens $\mathcal{L}_1 = \{waau \ / \ w, u \in \{a, b\}^*\}$ e $\mathcal{L}_2 = \{wbbbu \ / \ w, u \in \{a, b\}^*\}$. Os autômatos ilustrados na figura 4.3.A e 4.3.B, reconhecem as linguagem $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente. Note que $\mathcal{L}_1 = L((a+b)^*aa(a+b)^*)$ e $\mathcal{L}_2 = L((a+b)^*bbb(a+b)^*)$. A intersecção destas linguagens, isto é $\mathcal{L}_1 \cap \mathcal{L}_2$, é reconhecida pelo autômato ilustrado na figura 4.4.*

Definição 4.1.4 *Sejam Σ_1 e Σ_2 alfabetos. Uma função $h : \Sigma_1 \longrightarrow \Sigma_2^*$ é chamada um* **homomorfismo**.



(A)



(B)

Figura 4.3: AFD's que reconhecem as linguagens $\mathcal{L}_1 = L((a+b)^*aa(a+b)^*)$ e $\mathcal{L}_2 = L((a+b)^*bbb(a+b)^*)$.

Em outras palavras, um homomorfismo é uma substituição no qual um simples símbolo é trocado por uma cadeia. É possível estender a função h para uma função $\widehat{h}$ cujo domínio sejam cadeias em Σ_1 em vez de símbolos de modo óbvio.

Definição 4.1.5 *Sejam Σ_1 e Σ_2 alfabetos e $h : \Sigma_1 \longrightarrow \Sigma_2^*$ um homomorfismo. Um* **homomorfismo estendido** *de h, denotado por $\widehat{h}$ é a função $\widehat{h} : \Sigma_1^* \longrightarrow \Sigma_2^*$ definida por:*

1. $\widehat{h}(\lambda) = \lambda$
2. $\widehat{h}(a) = h(a)$ *para cada* $a \in \Sigma_1$
3. $\widehat{h}(wa) = \widehat{h}(w)h(a)$ *para cada* $w \in \Sigma_1^*$ *e* $a \in \Sigma_1$.

Assim, se $w = a_1a_2 \ldots a_n$, então $\widehat{h}(w) = h(a_1)h(a_2) \ldots h(a_n)$. Note que trivialmente $\widehat{h}(wv) = \widehat{w}\widehat{v}$.

Se $\mathcal{L}$ é uma linguagem, sobre Σ_1, sua **imagem homomorfa** é definida como

$$h(\mathcal{L}) = \{\widehat{h}(w) \ / \ w \in \mathcal{L}\}.$$

Note que $\widehat{h}(wv) = \widehat{w}\widehat{v}$. Portanto, trivialmente, temos que $h(\mathcal{L}_1\mathcal{L}_2) = h(\mathcal{L}_1)h(\mathcal{L}_2)$.

Por simplicidade notacional e em vista de que $\widehat{h}$ é uma extensão óbvia de h, de aqui em diante usaremos o mesmo nome de função (h) para ambos o homomorfismo e sua extensão.

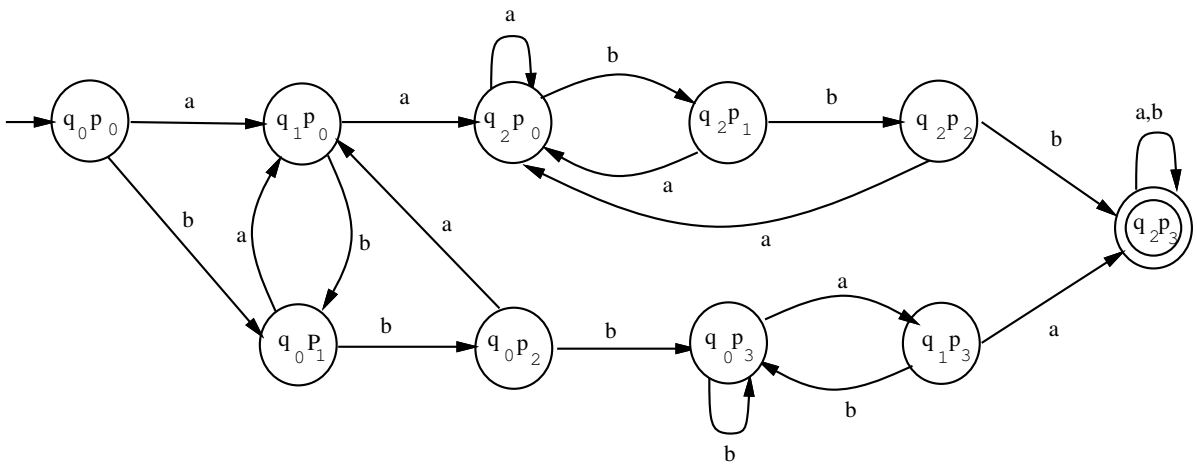


Figura 4.4: AFD que reconhece $L((a+b)^*aa(a+b)^*) \cap L((a+b)^*bbb(a+b)^*)$.

Exemplo 4.1.6 *Sejam* $\Sigma_1 = \{a, b\}$ *e* $\Sigma_2 = \{a, b, c\}$. *Defina* h *por* $h(a) = ab$ *e* $h(b) = bbc$. *Então* $h(aba) = h(a)h(b)h(a) = abbbcab$. *A imagem homomorfa de* $\mathcal{L} = \{aa, aba\}$ *é a linguagem*

$$h(\mathcal{L}) = \{abab, abbbcab\}.$$

Exemplo 4.1.7 *Tome* $\Sigma_1 = \{a, b\}$ *e* $\Sigma_2 = \{b, c, d\}$. *Defina* $h : \Sigma_1 \longrightarrow \Sigma_2^*$ *por*

$h(a) = dbcc$
$h(b) = bdc$

Se $\mathcal{L}$ *é a linguagem regular denotada por* $r = (ab + b^*)(aa)^*$, *então*

$$r_1 = (h(a)h(b) + h(b)^*)(h(a)h(a))^* = (dbccbdc + (bdc)^*)(dbccdbcc)^*,$$

denota a linguagem regular $h(\mathcal{L})$.

Teorema 4.1.8 *Seja* $h : \Sigma_1 \longrightarrow \Sigma_2^*$ *um homomorfismo. Se* $\mathcal{L}$ *é uma linguagem regular, sobre o alfabeto* Σ_1, *então sua imagem homomorfa é uma linguagem regular sobre o alfabeto* Σ_2. *A família das linguagens regulares é, portanto, fechada sob homomorfismos arbitrários.*

DEMONSTRAÇÃO: Seja $\mathcal{L}$ uma linguagem regular, sobre Σ_1, denotada por alguma expressão regular r, isto é $L(r) = \mathcal{L}$. Seja $h(r)$ a expressão obtida ao substituir em r, cada símbolo $a \in \Sigma_1$ por $h(a) \in \Sigma_2^*$, de modo similar ao exemplo 4.1.7. Note que no caso particular das expressões regulares primitivas λ e $\emptyset$, $h(\lambda) = \lambda$ e $h(\emptyset) = \emptyset$. Pode ser mostrado diretamente apelando à definição de uma expressão regular que o resultado $(h(r))$ é uma expressão regular. É fácil ver que a expressão regular $h(r)$ denota $h(L(r))$. Tudo que precisamos fazer é mostrar que para todo $w \in L(r)$, o correspondente $h(w)$ está em $L(h(r))$ e, inversamente, que para todo $v \in L(h(r))$

existe $w \in \mathcal{L}$ tal que $v = h(w)$. Noutras palavras, devemos mostrar que $L(h(r)) = h(L(r))$. Deixamos os detalhes como exercício. ■

A importância do homomorfismo, é que pode server para simplificar demonstrações. Por exemplo, se sabemos que a linguagem $\mathcal{L} = \{10^n1^k0 \ / \ n, k \geq 1\}$, então via o homomorfismo $h(0) = aa$ e $h(1) = bb$, saberemos que a linguagem $\mathcal{L}$' $= \{bb(aa)^n(bb)^kaa \ \ n, m$ são pares maiores ou iguais a $\{2\}$ também é uma linguagem regular, não precisamos construir um autômato finito que reconheça ela, ou uma gramática regular que a gere ou uma expressão regular que denote essa linguagem.

As linguagens regulares também são fechadas sobre reversões, prefixos e sufixos, como mostra o seguinte teorema.

Teorema 4.1.9 *Se $\mathcal{L}$ é uma linguagem regular, então $\mathcal{L}^R$, $\mathcal{L}^P$ e $\mathcal{L}^S$ também são regulares.*

Demonstração: Se $\mathcal{L}$ é regular, então existe uma gramática linear à direita que gera $\mathcal{L}$. Logo, o procedimento descrito no teorema 3.6.5 teríamos uma gramática linear à esquerda que gera $\mathcal{L}^R$ e portanto $\mathcal{L}^R$ é uma linguagem regular.

Se $\mathcal{L}$ é regular, então existe um AFD, M, que reconhece $\mathcal{L}$. Construa um novo AFD a partir de M dando o status de estado final a cada estado que esteja em algum caminho do estado inicial a algum estado final de M. Claramente, $L(M) = \mathcal{L}^P$.

Por outro lado, como trivialmente $\mathcal{L}^S = ((\mathcal{L}^R)^P)^R$, e linguagens regulares são fechadas sobre prefixos e reversões, então $\mathcal{L}^S$ é também uma linguagem regular. ■

4.2 Questões Elementares sobre Linguagens Regulares

Podemos discutir agora o seguinte problema fundamental: dada uma linguagem arbitrária $\mathcal{L}$ e uma cadeia w, podemos determinar algoritmicamente se w pertence a $\mathcal{L}$ ou não? Esta questão é conhecida como **questão de pertinência** e o método de respondê-la é o algoritmo de pertinência. A questão da existência e natureza desse algoritmo será de muito interesse no que seguirá. É um problema em geral difícil. Para as linguagens regulares, no entanto, ele é fácil.

O que queremos dizer por "dado uma linguagem ..."? Em muitos argumentos é importante que essa frase seja clara. Usamos várias maneiras de descrever as linguagens regulares: descrição verbal informal, notação de conjunto, autômatos finitos, expressões regulares e gramáticas regulares. Somente as três últimas são suficientemente bem definidas para usar em provas de teoremas. Portanto, dizemos que uma linguagem regular é dada numa **representação padrão** se, e somente se, ela é descrita por um autômato finito, uma expressão regular ou uma gramática regular.

Teorema 4.2.1 *Dado uma representação padrão de uma linguagem regular arbitrária $\mathcal{L}$, sobre Σ, e uma cadeia $w \in \Sigma^*$, existe um algoritmo para determinar se $w \in L$ ou não.*

Demonstração: Primeiro representamos a linguagem por algum autômato finito (se a representação padrão da linguagem é outra, então podemos, usando os algoritmo deste capítulo, sempre transformar este num autômato finito equivalente). Depois simulamos o comportamento

do autômato para a entrada w. Se ao finalizar a simulação o autômato pára num estado final, então o algoritmo aceita a cadeia senão o algoritmo rejeita a cadeia. ■

Existem diversos simuladores de autômatos finitos disponíveis na internet, por exemplo o SAGEMoLic desenvolvido pelo grupo de Teoria da Computação da Universidade de Brasilia e disponível em: http://www.mat.unb.br/ ayala/TCgroup/SAGEMoLiC/.

Outras questões importantes são:

1. $\mathcal{L}$ é uma linguagem vazia ou não vazia?
2. $\mathcal{L}$ é uma linguagem finita ou infinita?
3. São $\mathcal{L}_1$ e $\mathcal{L}_2$ a mesma linguagem?
4. $\mathcal{L}_1$ é um subconjunto de $\mathcal{L}_2$?

Teorema 4.2.2 *Existe um algoritmo para determinar se uma dada linguagem regular, na representação padrão, é vazia, finita ou infinita.*

Demonstração: A resposta é imediata a partir da representação da linguagem como grafo de transição de um AFD. Se existe um caminho do vértice inicial a qualquer vértice final, então a linguagem é não-vazia.

Para determinar se a linguagem é infinita ou não, achamos todos os vértices que são a base de algum ciclo. Se alguns desses estão sobre um caminho de um vértice inicial a um final, a linguagem é infinita. Caso contrário, ela é finita. ■

Algoritmos para encontrar caminhos e ciclos são bem conhecidos em teoria dos grafos.

Outra forma de verificar se uma linguagem é infinita, é verificar se na representação padrão de expressões regulares, ela contém o fecho estrela $*$ aplicado a uma expressão contendo pelo menos um símbolo de Σ.

A questão da igualdade de duas linguagens é também um problema prático importante. Freqüentemente, existem várias definições de uma linguagem de programação e, precisamos saber, apesar de suas diferenças aparente, se elas especificam a mesma linguagem. Esse, em geral, é um problema difícil. Mesmo para linguagens regulares o argumento não é óbvio. Não é possível argumentar na comparação palavra a palavra, pois ele só funciona para linguagens finitas. Não é também fácil apelar para expressões regulares, gramáticas regulares ou AFD's, pois existem, por exemplo, infinitas expressões regulares para denotar a mesma linguagem. Uma solução elegante usa a propriedade de fecho.

Teorema 4.2.3 *Dadas as linguagens regulares $\mathcal{L}_1$ e $\mathcal{L}_2$, na representação padrão, existe um algoritmo para determinar se $\mathcal{L}_1 = \mathcal{L}_2$ ou não.*

DEMONSTRAÇÃO: Usando $\mathcal{L}_1$ e $\mathcal{L}_2$ podemos construir a linguagem

$$\mathcal{L}_3 = (\mathcal{L}_1 \cup \mathcal{L}_2) - (\mathcal{L}_1 \cap \mathcal{L}_2).$$

$\mathcal{L}_3$ é regular, pois $\overline{\mathcal{L}_1}$ e $\overline{\mathcal{L}_2}$ são e pelo teorema 4.1.1, a união, intersecção e diferença de duas linguagens regulares são regulares. Portanto, podemos achar um AFD, M, que reconheça $\mathcal{L}_3$. Observe que se $\mathcal{L}_1 = \mathcal{L}_2$, então

$$\begin{aligned}\mathcal{L}_3 &= (\mathcal{L}_1 \cup \mathcal{L}_2) - (\mathcal{L}_1 \cap \mathcal{L}_2)\\ &= (\mathcal{L}_1 \cup \mathcal{L}_1) - (\mathcal{L}_1 \cap \mathcal{L}_1)\\ &= \mathcal{L}_1 - \mathcal{L}_1\\ &= \emptyset\end{aligned}$$

Se $\mathcal{L}_1 \neq \mathcal{L}_2$, então ou existe um elemento $w \in \mathcal{L}_1$ tal que $w \notin \mathcal{L}_2$ ou existe um elemento $w \in \mathcal{L}_2$ tal que $w \notin \mathcal{L}_1$. Em ambos os casos $w \in \mathcal{L}_1 \cup \mathcal{L}_2$ e $w \notin \mathcal{L}_1 \cap \mathcal{L}_2$. Logo, $\mathcal{L}_3 = (\mathcal{L}_1 \cup \mathcal{L}_2) - (\mathcal{L}_1 \cap \mathcal{L}_2) \neq \emptyset$.

Assim, $\mathcal{L}_1 = \mathcal{L}_2$ se, e somente se, $\mathcal{L}_3 = \emptyset$.

Logo, podemos usar o algoritmo do teorema 4.2.2 para determinar se $\mathcal{L}_3$ é vazia ou não e assim concluir se $\mathcal{L}_1 = \mathcal{L}_2$ ou $\mathcal{L}_1 \neq \mathcal{L}_2$. ■

4.3 Identificando Linguagens Não Regulares

As linguagens regulares podem ser infinitas. No entanto, o fato delas poderem ser associados com autômatos que tem memória finita, impõe alguns limites na estrutura das linguagens regulares. Nossa intuição diz que uma linguagem é regular somente se, em processando qualquer cadeia, a informação a ser armazenada em qualquer estágio é estritamente limitada. Isso é verdade, mas teremos de mostrar precisamente.

4.3.1 Usando o Princípio da Casa de Pombos

O termo "princípio da casa de pombos" é usado pelos matemáticos para se referir à seguinte simples observação: Se colocamos n objetos (pombos) em m caixas (casa de pombos), e se $n > m$, então no mínimo uma caixa deve conter mais de um ítem. Nesta metáfora, a casa dos pombos correspondem aos estados de um AFD e os n pombos a uma cadeia de tamanho n.

Exemplo 4.3.1 *A linguagem $\mathcal{L} = \{a^n b^n / \ n \geq 0\}$ é regular? A resposta é não. Mostraremos usando uma prova por contradição. Suponha que $\mathcal{L}$ é regular. Então, existe um afd $M = \langle Q, \{a, b\}, \delta, q_0, F\rangle$ que a reconhece. Agora, olhemos para $\delta^*(q_o, a^i)$, com $i = 1, 2, 3, \ldots$. Como existe um número ilimitado de i's, mas somente um número limitado de estados em M, o princípio da casa de pombos nos diz que para $n > |Q|$ deve existir algum estado, digamos q, tal que*

$$\delta^*(q_0, a^n) = q \text{ e } \delta^*(q_0, a^m) = q, \text{ para algum } m \neq n.$$

Mas, como M aceita $a^n b^n$, devemos ter

$$\begin{aligned} \delta^*(q_0, a^n b^n) &= q_f \in F \\ \delta^*(\delta^*(q_0, a^n), b^n) &= q_f \in F \\ \delta^*(q, b^n) &= q_f \in F. \end{aligned}$$

Disso podemos concluir que

$$\delta^*(q_0, a^m b^n) = \delta^*(\delta^*(q_0, a^m), b^n) = \delta^*(q, b^n) = q_f \in F.$$

Isto contradiz a suposição original de que M aceita $a^m b^n$ somente se $m = n$ (e neste caso $m \neq n$). Logo, $\mathcal{L}$ não pode ser regular.

Neste argumento, o princípio da casa de pombos é uma maneira de estabelecer precisamente o que queremos dizer quando dizemos que um autômato finito tem memória limitada. Para reconhecer todos $a^n b^n$, um autômato teria de distinguir todos os prefixos de a^n. Mas como existe somente um número finito de estados internos para fazer isso, quando n for maior que a quantidade de estados existiriam alguns prefixos de a^n para os quais essa distinção não poderia ser feita.

Para usar esse tipo de argumento numa variedade de situações é conveniente codificá-lo como um teorema geral. Existem várias maneiras de fazer isso, a que faremos é talvez a mais famosa.

4.3.2 Lema do Bombeamento para Linguagens Regulares

O resultado que segue, conhecido como *lema do bombeamento* para linguagens regulares, usa o princípio da casa de pombos numa outra forma. A prova é baseada na observação de que num grafo de transição com n vértices, qualquer caminho de comprimento n ou mais longo deve repetir algum vértice, isto é, contém um ciclo. O lema do bombeamento só precisará analisar linguagens infinitas, pois trivialmente toda linguagem finita necessariamente é regular.

Teorema 4.3.2 (Lema do bombeamento) *Seja $\mathcal{L}$ uma linguagem infinita. Se $\mathcal{L}$ é regular então, existe um inteiro positivo m tal que todo $w \in \mathcal{L}$, com $|w| \geq m$, pode ser decomposto como $w = xyz$, com $|xy| \leq m$ e $|y| \geq 1$ tal que*

$$w_i = xy^i z, \tag{4.1}$$

está também em $\mathcal{L}$, para todo $i = 0, 1, 2, \ldots$.

DEMONSTRAÇÃO: Se $\mathcal{L}$ é regular, existe um AFD que a reconhece. Seja M um desses AFD's, com estados rotulados por $q_0, q_1, \ldots, q_n$. Agora tome uma cadeia $w \in L$ tal que $|w| = k \geq m = n+1$. Como $\mathcal{L}$ é infinita isso pode sempre ser considerado. Seja $q_0, q_i, q_j, \ldots, q_f$ o conjunto de estados do autômato quando ele reconhece w.

Como essa cadeia tem no mínimo $n + 1$ entradas, então pelo menos um estado deve ser repetido, e tal repetição deve começar não após o m-ésimo movimento. Portanto, a seqüência deve ter a seguinte forma

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

indicando que devem existir subcadeias x, y, e z de w tal que

$\delta^*(q_0, x) = q_r$,
$\delta^*(q_r, y) = q_r$,
$\delta^*(q_r, z) = q_f$,

com $|xy| \leq n + 1 = m$ e $|y| \geq 1$. Donde segue imediatamente que

$$\delta^*(q_0, xz) = q_f, \text{ assim como } \delta^*(q_0, xy^2z) = q_f, \;\; \delta^*(q_0, xy^3z) = q_f,$$

e assim por diante, completando a prova do teorema. ■

Ou seja, se $\mathcal{L}$ for regular e infinita, então toda palavra suficientemente longa em $\mathcal{L}$ pode ser quebrada em três partes, de tal modo que um número arbitrário de repetições da parte do meio gera necessariamente uma outra palavra em $\mathcal{L}$. Dizemos que a cadeia do meio é "bombeada". mas isto não significa que em qualquer decomposição xyz da cadeia w, y possa ser bombeada arbitrariamente, ou seja perfeitamente podem existir decomposições de w tais que o bombeamento da parte do meio gere uma cadeia não pertencente à linguagem.

Por exemplo a linguagem $\mathcal{L} = \{a^n \ / \ n \text{ é par}\}$ é infinita e regular e portanto satisfaz o lema do bombeamento. Para ver isto basta escolher $m = 2$, pois qualquer cadeia $w = a^{2k}$ para algum $k \geq 1$ pode ser decomposta em $x = \lambda$, $y = aa$ e $z = a^{2(k-1)}$, ao bombear y i-vezes, teremos a cadeia

$$xy^iz = (aa)^i a^{2(k-1)} = a^{2i}a2(k-1) = a^{2(i+(k-1))}$$

que é da forma $a^{2k'}$ e portanto faz parte da linguagem $\mathcal{L}$. Observe que se tivéssemos decomposto w em $x = \lambda$, $y = a$ e $z = a^{2k-1}$, ao bombear y duas vezes ($i = 2$), teríamos a cadeia $xy^2z = a^2a^{2k-1} = a^{2+2k-1} = a^{2k+1}$ que não é de comprimento par.

Enunciamos o lema do bombeamento somente para linguagens infinitas. Linguagens finitas, embora sempre regulares, não podem ser bombeadas pois o bombeamento automaticamente cria um conjunto infinito de cadeias. Assim, o teorema não vale para linguagens finitas, pois ele se tornaria vácuo (poderia ser escolhido um m maior do que o maior comprimento das cadeias na linguagem, de modo que nenhuma cadeia pode ser bombeada).

Exemplo 4.3.3 *Seja a linguagem*

$$\mathcal{L} = \{w \in \{a, b\}^* \ / \ \mathcal{N}_a(w) \text{ e } \mathcal{N}_b(w) \text{ são pares}\}.$$

Esta linguagem é regular (veja exemplo 2.5). Logo, pelo lema do bombeamento deve existir um inteiro positivo m tal que todo $w \in \mathcal{L}$, com $|w| \geq m$, pode ser decomposto como $w = xyz$, com $|xy| \leq m$ e $|y| \geq 1$ satisfazendo a equação 4.1.

Seja $m = 3$ e uma cadeia arbitrária $w = a_1a_2 \dots a_n \in \mathcal{L}$ (Portanto $\mathcal{N}_a(w) = 2p$ e $\mathcal{N}_b(w) = 2q$ para algum $p, q \in \mathbb{N}$) tal que $|w| = n \geq m = 3$. Claramente, como so temos dois símbolos no alfabeto ou $a_2 = a_1$ ou $a_2 = a_3$. Supondo que $a_2 = a_1 = a$ então podemos decompor w em $x = \lambda$, $y = a_1a_2$ e $z = a_3 \dots a_n$, bombeando y k vezes, nos teremos que

$$\begin{aligned}\mathcal{N}_a(xy^kz) &= \mathcal{N}_a(x) + \mathcal{N}_a(y^k) + \mathcal{N}_a(z)\\ &= 0 + \mathcal{N}_a((aa)^k) + (\mathcal{N}_a(w) - 2)\\ &= 2k + 2p - 2\\ &= 2(k + p - 1)\end{aligned}$$

Portanto, $\mathcal{N}_a(xy^kz)$ é par. por outro lado, como não modificamos a quantidade de $b's$, temos que $\mathcal{N}_b(xy^kz) = \mathcal{N}_b(w) = 2q$. Assim, $xy^kz \in \mathcal{L}$. Se $a_2 = a_1 = b$ procedemos de forma similar.

Já se $a_2 = a_3 = a$ ou se $a_2 = a_3 = b$ então podemos decompor w em $x = a_1$, $y = a_2a_3$ e $z = a_4 \dots a_n$ e proceder de forma análoga ao caso anterior.

Portanto $\mathcal{L}$ é uma linguagem regular.

Porém o lema do bombeamento, assim como o argumento do princípio da casa de pombos do exemplo 4.3.1, é mais usado para se mostrar que certas linguagens não são regulares. A demonstração é sempre por contradição.

Exemplo 4.3.4 *Usaremos o lema do bombeamento para mostrar que $\mathcal{L} = \{a^nb^n / \ n \geq 0\}$ não é uma linguagem regular. Suponhamos que $\mathcal{L}$ é regular, portanto o lema do bombeamento deveria valer. Não sabemos o valor de m, mas qualquer que seja ele podemos escolher $n > m$. Logo, a subcadeia y deve consistir inteiramente de a's. Suponha que $|y| = k$ com $k \geq 1$. Então, a cadeia obtida usando-se $i = 0$, na equação 4.1, é*

$$a^{n-k}b^n.$$

Como $n - k \neq n$, então, claramente, $a^{n-k}b^n$ não está em $\mathcal{L}$. Isso contradiz o lema do bombeamento e, portanto, a hipótese é falsa, isto é, $\mathcal{L}$ não é regular.

Exemplo 4.3.5 *Seja a linguagem $\mathcal{L}_{Pal}$ do exemplo 3.5.7. Vamos demonstrar usando o lema do bombeamento, que esta linguagem não é regular. Suponhamos que é regular, portanto o lema do bombeamento deveria valer. Como não conhecemos o valor de m, usemos um m genérico. Agora escolha a cadeia*

$$a^mba^m.$$

Claramente, a^mba^m é um palíndromo, independente do valor de m. Pelo lema do bombeamento, podemos decompor essa cadeia em três partes (x, y e z) satisfazendo as condições do lema do bombeamento e bombear a subcadeia y tantas vezes quanto desejarmos, que ainda obtemos uma cadeia em $\mathcal{L}_{Pal}$. Para refutar este resultado devemos mostrar que qualquer que seja esta decomposição de a^mba^m (satisfazendo as condições do lema) podemos gerar, ao bombear a subcadeia y, uma cadeia que não pertence à linguagem. Faremos estas decomposições genericamente. Como $|xy| \leq m$ e $|y| \geq 1$ então $x = a^i$ com $i \leq m$, $y = a^j$ com $1 \leq j \leq m$, $i + j \leq m$

e $z = a^{m-(i+j)}ba^m$. *Pelo lema do bombeamento, todas as cadeias da forma* xy^kz, *com* $k \in \mathbb{N}$ *também pertencem a* $\mathcal{L}_{Pal}$. *Porém para* $k = 2$, *temos que*

$$\begin{aligned} xy^2z &= a^ia^ja^ja^{m-(i+j)}ba^m \\ &= a^ia^ja^{m-(i+j)}a^jba^m \\ &= a^ma^jba^m \\ &= a^{m+j}ba^m \end{aligned}$$

e portanto $xy^2z \neq a^nb^{2n}a^{n+j})$ *para todo* $n \in \mathbb{N}$. *Ou seja* xy^2z *não é um palíndromo o que contradiz o lema do bombeamento. Logo a hipótese que* $\mathcal{L}_{Pal}$ *é uma linguagem regular é falsa.*

Estes exemplos sugerem que é possível reescrever o lema do bombeamento na sua forma contra-positiva para demonstrar diretamente quando uma linguagem não é regular.

Corolário 4.3.6 *Seja* $\mathcal{L}$ *uma linguagem infinita. Se para qualquer inteiro positivo* m *existe uma cadeia* $w \in \mathcal{L}$ *tal que* $|w| \geq m$ *e para toda possível decomposição de* w *em três cadeias* x, y *e* z $(w = xyz)$ *com* $|xy| \leq m$ *e* $|y| \geq 1$, *temos que a cadeia* $xy^iz \notin \mathcal{L}$ *para algum inteiro não negativo* i, *então* $\mathcal{L}$ *não é regular.*

Exemplo 4.3.7 *Demonstremos usando o corolário acima que a linguagem*

$$\mathcal{L} = \{uu \ / \ u \in \{a, b\}^*\}$$

não é regular.

Seja m *um inteiro positivo qualquer e* $w = a^mba^mb$. *Então,* $w = uu$ *para* $u = a^mb$ *e portanto* $w \in \mathcal{L}$ *e* $|w| = 2m + 2 > m$. *Suponha que* $w = xyz$ *com* $|xy| \leq m$ *e* $|y| \geq 1$. *Claramente,* y *está na primeira metade e está so composto de* $a's$. *No menor dos casos* $y = a$. *Logo se bombearmos* y, *digamos em três vezes, a quantidade de* $a's$ *da primeira metade aumentará pelo menos em dois, e portanto o único b que fazia parte da primeira metade passará para a segunda metade, resultando assim numa cadeia cuja metade esquerda é necessariamente diferente da metade direita e portanto não pertencerá à linguagem. Formalmente, se* $|x| = p$ *e* $|y| = q$ *(com* $q \geq 1$ *e* $p + q \leq m$*) então* $w = a^pa^qa^{m-(p+q)}ba^mb$. *Bombeando duas vezes* y *teremos a cadeia*

$$\begin{aligned} xy^3z &= a^pa^{2q}a^{m-(p+q)}ba^mb \\ &= a^{m-(p+q)+p+q+q}ba^mb \\ &= a^{m+q}ba^mb \end{aligned}$$

Como $q \geq 1$ *então* $m + q > m$ *e portanto* $xy^3z = a^{m+q}ba^mb \notin \mathcal{L}$. *Logo,* $\mathcal{L}$ *não é uma linguagem regular.*

Lamentavelmente, o lema do bombeamento estabelece apenas uma condição necessária para uma linguagem infinita ser regular. Mas isto não é suficiente para provarmos que uma linguagem qualquer seja regular ou não. Analogamente, o corolário 4.3.6, da uma condição suficiente, mas não necessária, para provarmos que uma linguagem não é regular, ou seja existem linguagens que satisfazem as condições desse corolário mas que não são regulares.

Exemplo 4.3.8 *Seja a linguagem*

$$\mathcal{L} = \{uu^R v \ / \ u, v \in \{a, b\}^+\}$$

Mostraremos que esta linguagem embora claramente não seja regular, não pode ser provada pelo lema do bombeamento.

Seja $m = 4$ *e* $w = a_1 \dots a_n \in \mathcal{L}$ *tal que* $n \geq 4$. *Por definição da linguagem, existe um número* $1 \leq k \leq \lceil \frac{n}{2} \rceil - 1$ *tal que para cada* $1 \leq j \leq k$, $a_j = a_{2k-j+1}$, *ou seja fazendo* $u = a_1 \dots a_k$ *e* $v = a_{2k+1} \dots n$ *temos que* $|u| \geq 1$, $|v| \geq 1$ *e* $w = uu^R v$.

Se $k = 1$ *então é suficiente escolher* $x = a_1 a_2$, $y = a_3$ *e* $z = a_4 \dots a_n$, *pois neste caso* $x = uu^R$ *e portanto trivialmente* $xy^i z \in \mathcal{L}$.

Se $k \geq 2$ *então considere* $x = \lambda$, $y = a_1$ *e* $z = a_2 \dots a_n$. *Note que se* $a_1 \dots a_{2k}$ *é uma palíndromo de tamanho par, então* $a_2 \dots a_{2k-1}$ *também é um palíndromo de tamanho par. Logo, para* $i = 0$, $xy^i z = z = a_2 \dots a_n = u' u'^R v$, *onde* $u' = a_2 \dots a_k$ *e* $v = a_{2k+1} \dots a_n$ *e por tanto está em* $\mathcal{L}$. *Se* $i = 1$ *então trivialmente* $xy^i z = xyz = w \in \mathcal{L}$. *Se* $i \geq 2$ *então* $xy^i z = a_1 a_1 a_1^{i-2} a_2 \dots a_n$. *Assim tomando* $u = a_1$ *e* $v = a_1^{i-2} a_2 \dots a_n$ *temos que* $|u| \geq 1$, $|v| \geq 1$ *e* $w = uu^R v$. *Portanto* $xy^i z \in \mathcal{L}$.

Desta forma seria impossível aplicar o corolário 4.3.6 para provarmos que essa linguagem não é regular.

Existem formas seguras de provarmos que linguagens não regulares são de fato não regulares. Por exemplo [LV95] apresenta uma técnica eficiente com provas simples, mas que estão baseadas na complexidade de Kolmogorov que não foi considerada neste texto.

4.4 Exercícios

1. Prove que toda linguagem finita é regular.

2. Aplique a construção usada na prova do teorema 4.1.1 para obter o AFD que reconhece a intersecção das linguagens aceitas pelos AFD's da figura 4.5.

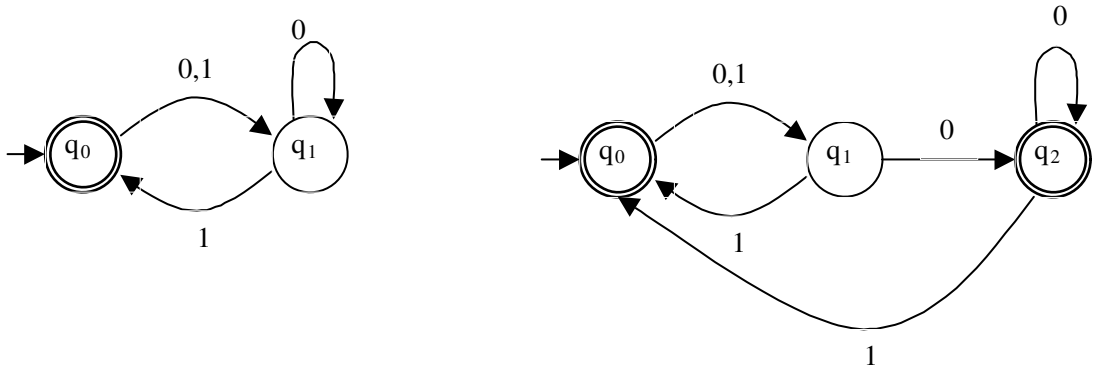


Figura 4.5: AFD do exercício 2.

3. Quais das seguintes igualdades são verdadeiras para todas as linguagens regulares e todos os homomorfismos? Justifique.

   (a) $h(\mathcal{L}_1 \cup \mathcal{L}_2) = h(\mathcal{L}_1) \cup h(\mathcal{L}_2)$
   (b) $h(\mathcal{L}_1 \cap \mathcal{L}_2) = h(\mathcal{L}_1) \cap h(\mathcal{L}_2)$
   (c) $h(\mathcal{L}^n) = h(\mathcal{L})^n$
   (d) $h(\mathcal{L}^*) = h(\mathcal{L})^*$
   (e) $h(\mathcal{L}^R) = h(\mathcal{L})^R$
   (f) $h(\overline{\mathcal{L}}) = \overline{h(\mathcal{L})}$
   (g) $h(\mathcal{L}_1 - \mathcal{L}_2) = h(\mathcal{L}_1) - h(\mathcal{L}_2)$

4. Na prova do teorema 4.1.8, mostre que $h(r)$ é uma expressão regular. Então, mostre que $h(r)$ denota $h(\mathcal{L})$.

5. Mostre que a família das linguagens regulares é fechada sobre união finita e intersecção finita, isto é, se $\mathcal{L}_1$, $\mathcal{L}_2, \ldots, \mathcal{L}_n$ são linguagens regulares então

$$\bigcup_{i=1}^{n} \mathcal{L}_i \quad \text{e} \quad \bigcap_{i=1}^{n} \mathcal{L}_i$$

   também são.

6. Sejam $\mathcal{L}_1$ e $\mathcal{L}_2$ duas linguagens sobre os alfabetos Σ_1 e Σ_2, respectivamente. O NEM de $\mathcal{L}_1$ e $\mathcal{L}_2$ é definido por

$$NEM(\mathcal{L}_1, \mathcal{L}_2) = \{w \in (\Sigma_1 \cup \Sigma_2)^* \ / \ w \notin \mathcal{L}_1 \text{ e } w \notin \mathcal{L}_2\}.$$

   Mostre que a família das linguagens regulares é fechada sobre NEM.

7. Mostre que as linguagens regulares são fechadas sobre a união disjunta, onde a união disjunta de dois conjuntos (linguagens) $\mathcal{L}_1$ e $\mathcal{L}_2$ é definida por

$$\mathcal{L}_1 \uplus \mathcal{L}_2 = \{w0/ \ w \in \mathcal{L}_1\} \cup \{w1/ \ w \in \mathcal{L}_2\}$$

8. Seja $\Sigma = \{a, b\}$ e $\widehat{\ }: \Sigma^* \to \Sigma^*$ a operação definida recursivamente a seguir:

   - $\widehat{\lambda} = a$
   - $\widehat{wa} = b\widehat{w}$
   - $\widehat{wb} = a\widehat{w}$

   Seja $\mathcal{L}$ uma linguagem regular sobre Σ. Mostre que $\widehat{\mathcal{L}} = \{\widehat{w}/ \ w \in \Sigma^*\}$ também é regular.

9. Seja $\Sigma = \{a, b\}$ e $\widehat{\ }: \Sigma^* \to \Sigma^*$ a operação definida a seguir:

$$\widehat{\mathcal{L}} = \{wa \ / \ wb \in \mathcal{L}\} \cup \{wb \ / \ wa \in \mathcal{L}\}$$

   Mostre que a classe das linguagens regulares é fechada sobre esta operação.

10. Seja $\mathcal{L}$ uma linguagem sobre um alfabeto Σ. Conforme visto no capítulo 1, a linguagem de **sufixos** de $\mathcal{L}$, denotada por $\mathcal{L}^S$, é definida como

$$\mathcal{L}^S = \{w \in \Sigma^* \ / \ vw \in \mathcal{L} \text{ para algum } v \in \Sigma^*\}.$$

    Demonstre que a classe das linguagens regulares é fechada sobre sufixos.

11. Seja $\mathcal{L}$ uma linguagem sobre um alfabeto Σ. Defina

    (a) $L^{\ominus} = \{w/w \notin L \text{ e } |w| \neq 100\}$
    (b) $\mathcal{L}^T = \{w \in \Sigma^* \ / \ aw \in \mathcal{L} \text{ para algum } v \in \Sigma^*\}$. Observe que claramente, $\mathcal{L}^T \subseteq \mathcal{L}^S$.

    Demonstre que a classe das linguagens regulares é fechada sobre estes operadores.

12. Sejam $\mathcal{L}_1$ e $\mathcal{L}_2$ duas linguagens, defina o operador $\oplus$ por

$$\mathcal{L}_1 \oplus \mathcal{L}_2 = \{w \in \mathcal{L}_1/ \ w \notin \mathcal{L}_2\} \cup \{w \in \mathcal{L}_2/ \ w \notin \mathcal{L}_1\}$$

    Mostre que as linguagens regulares são fechadas sobre a operação $\oplus$ entre linguagens.

	a	b	c
a	a	a	c
b	c	a	b
c	b	c	a

Tabela 4.1: Tabela multiplicativa não associativa nem comutativa.

13. Sejam $\mathcal{L}_1$ e $\mathcal{L}_2$ linguagens sobre alfabetos Σ_1 e Σ_2, respectivamente. Defina o operador $\wedge_2$ por

$$\mathcal{L}_1 \wedge_2 \mathcal{L}_2 = \{uv/\ u \in \mathcal{L}_1 \cup \mathcal{L}_2 \text{ e } v \in (\Sigma_1 \cup \Sigma_2)^*\}$$

Mostre que as linguagens regulares são fechadas sobre a operação $\wedge_2$ entre linguagens.

14. Seja $\Sigma = \{a_0, \ldots, a_n\}$. Defina o complemento n de $w \in \Sigma^*$, denotado por w^c, como sendo $\lambda^c = \lambda$ e $(wa_i)^C = w^c a_{n-i}$

Por exemplo se $\Sigma = \{a, b, c, d\}$ e considerando $a_0 = a$, $a_1 = b$, $a_2 = c$ e $a_3 = d$, então $aabcd^c = ddcba$.

Se $\mathcal{L}$ é uma linguagem regular sobre um alfabeto $\Sigma = \{a_0, \ldots, a_n\}$, demonstre que a linguagem

$$\mathcal{L}^c = \{w^c\ /\ w \in \mathcal{L}\},$$

também é regular.

15. Mostre que existe um algoritmo para determinar se $\mathcal{L}_1 \subseteq \mathcal{L}_2$, para qualquer linguagem regular $\mathcal{L}_1$ e $\mathcal{L}_2$.

16. Mostre que existe um algoritmo para determinar se a intersecção e união de duas linguagens regulares é finita, vazia ou infinita.

17. Seja a tabela 4.1. A primeira coluna representa os rótulos de cada fila na tabela enquanto a primeira fila representa os rótulos de cada coluna na tabela. Para cada $x, y \in \Sigma = \{a, b, c\}$ denote por $T(x, y)$ o conteúdo da tabela para a posição (x, y). Seja a seguinte função $A : \Sigma^+ \to \Sigma$ definida por

    - $A(a) = a$, para cada $a \in \Sigma$ e
    - $A(wa) = T(A(w), a)$, para cada $w \in \Sigma^+$ e $a \in \Sigma$.

    Demonstre que a seguinte linguagem é regular:

$$\mathcal{L} = \{w \in \Sigma^+ / A(w) = A(w^R)\}$$

18. Seja $\Sigma = \{0, 1\}$. Mostre que é possível aplicar o lema do bombeamento para as seguintes linguagens regulares.

(a) $\mathcal{L} = \{w \in \Sigma^* \ / \ 0110 \text{ é um prefixo de } w\}$
(b) $\mathcal{L} = \{w \in \Sigma^* \ / \ 0110 \text{ é um sufixo de } w\}$
(c) $\mathcal{L} = \{w \in \Sigma^* \ / \ w = u111v \text{ para algum } u, v \in \Sigma^*\}$
(d) $\mathcal{L} = \{w \in \Sigma^* \ / \ \nexists u, v \in \Sigma^* \text{ tal que } w = u111v\}$

19. Mostre que as linguagens sobre $\Sigma = \{a, b\}$, definidas a seguir, não são regulares.

(a) $\mathcal{L} = \{w \in \Sigma^* / \ \mathcal{N}_a(w) = \mathcal{N}_b(w)\}$
(b) $\mathcal{L} = \{w \in \Sigma^* / \ \mathcal{N}_a(w) \neq \mathcal{N}_b(w)\}$
(c) $\mathcal{L} = \{w \in \Sigma^* / \ u = v \text{ para algum prefixo } u \neq \lambda \text{ e sufixo } v \text{ de } w\}$.
(d) $\mathcal{L} = \{a^m b^n / \ m > n\}$
(e) $\mathcal{L} = \{a^m b^n / \ m \neq n\}$
(f) $\mathcal{L} = \{a^n b a^{n+1} / \ n \geq 1\}$
(g) $\mathcal{L} = \{a^m b^n / \ 1 \leq m \leq n \leq 2m\}$
(h) $\mathcal{L} = \{a^{m+1} b^{n+1} / \ 2 \leq n \leq m \leq 3n\}$

onde $\mathcal{N}_a(w)$ é o número de a's que figuram em w

20. Seja $\Sigma = \{a, b\}$ e $\widehat{\ }: \Sigma^* \to \Sigma^*$ a operação definida recursivamente a seguir:

- $\widehat{\lambda} = \lambda$
- $\widehat{wa} = \widehat{w}b$
- $\widehat{wb} = \widehat{w}a$

Mostre usando o lema do bombeamento que as linguagens

(a) $\mathcal{L} = \{w\widehat{w} / \ w \in \Sigma^*\}$
(b) $\mathcal{L} = \{w \in \Sigma^* / \ w = \widehat{w}^R\}$

não são linguagens regulares.

21. Seja $\Sigma = \{a, b\}$ e $\widehat{\ }: \Sigma^* \to \Sigma^*$ a operação definida recursivamente a seguir:

- $\widehat{\lambda} = \lambda$
- $\widehat{wa} = \widehat{w}aa$
- $\widehat{wb} = \widehat{w}bb$

Mostre usando o lema do bombeamento que a linguagem $\mathcal{L} = \{w\widehat{w} / \ w \in \Sigma^*\}$ não é regular.

22. Seja $\mathcal{L}$ uma linguagem sobre um alfabeto Σ qualquer. Defina $P(\mathcal{L})$, recursivamente por

- $P(\lambda) = \lambda$
- $P(wa) = aP(w)aa$ para cada $a \in \Sigma$ e $w \in \Sigma^*$.

Mostre usando o lema do bombeamento que a linguagem $\mathcal{L} = \{P(w) / \ w \in \Sigma^*\}$ não é regular.