

**Exercício 1.1.** Prove por indução em  $n$  que:

$$a) \sum_{i=0}^n i = \frac{n(n+1)}{2}$$

$$b) \sum_{i=0}^n i^3 = \left( \sum_{i=0}^n i \right)^2$$

**Solução:**

a) **Demonstração.** Por indução em  $n$ .

*Base:* Fazendo  $n = 0$  e calculando ambos os lados da expressão, temos:  $\sum_{i=0}^n i = \sum_{i=0}^0 i = 0$  e  $\frac{n(n+1)}{2} = \frac{0(0+1)}{2} = 0$ . Portanto, a expressão vale para  $n = 0$ .

*Hipótese Indutiva:* Por definição, temos:  $\sum_{i=0}^{n+1} i = \sum_{i=0}^n i + (n+1)$ . Usando a hipótese indutiva, isto é, supondo que a expressão vale para  $n$ , temos:  $\sum_{i=0}^n i + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{n(n+1) + 2(n+1)}{2} = \frac{(n+1)(n+2)}{2} = \frac{(\mathbf{n}+1)(\mathbf{n}+1+1)}{2}$ . Portanto, a expressão vale para  $n+1$ .

Assim, pelo Princípio da Indução Finita (PIF), a expressão  $\sum_{i=0}^n i = \frac{n(n+1)}{2}$  vale para todo  $n \in \mathbb{N}$ .  $\square$

b) Do item (a) acima temos que  $\sum_{i=0}^n i = \frac{n(n+1)}{2}$ . Logo,  $\left( \sum_{i=0}^n i \right)^2 = \left( \frac{n(n+1)}{2} \right)^2$ .

Portanto, iremos provar, por indução em  $n$ , que  $\sum_{i=0}^n i^3 = \left( \frac{n(n+1)}{2} \right)^2$ .

**Demonstração.** Por indução em  $n$ .

*Base:* De fato,  $n = 0 \Rightarrow \sum_{i=0}^n i^3 = \sum_{i=0}^0 i^3 = 0^3 = 0$  e  $\left( \frac{n(n+1)}{2} \right)^2 = \left( \frac{0(0+1)}{2} \right)^2 = 0^2 = 0$ . Portanto, a expressão vale para  $n = 0$ .

*Hipótese Indutiva:* Por definição, temos:  $\sum_{i=0}^{n+1} i^3 = \sum_{i=0}^n i^3 + (n+1)^3$ . Usando a hipótese indutiva, temos:

$$\begin{aligned} \sum_{i=0}^n i^3 + (n+1)^3 &= \left( \frac{n(n+1)}{2} \right)^2 + (n+1)^3 = \frac{n^2(n+1)^2 + 4(n+1)^3}{4} = \\ &= \frac{n^2(n^2 + 2n + 1) + 4(n^3 + 3n^2 + 3n + 1)}{4} = \frac{n^4 + 2n^3 + n^2 + 4n^3 + 12n^2 + 12n + 4}{4} = \end{aligned}$$

$$\begin{aligned}
&= \frac{n^4 + 4n^3 + 4n^2 + 2n^3 + 8n^2 + 8n + n^2 + 4n + 4}{4} = \frac{(n^2 + 2n + 1)(n^2 + 4n + 4)}{4} = \\
&= \frac{(n+1)^2(n+2)^2}{4} = \left( \frac{(n+1)(n+2)}{2} \right)^2 = \left( \frac{(\mathbf{n}+1)(\mathbf{n}+1+1)}{2} \right)^2
\end{aligned}$$

Portanto, a expressão vale para  $n + 1$ .

Assim, pelo PIF, a expressão  $\sum_{i=0}^n i^3 = \left( \frac{n(n+1)}{2} \right)^2$  vale para todo  $n \in \mathbb{N}$ .

□

**Exercício 1.3.** Um palíndromo pode ser definido como uma cadeia que resulta no mesmo valor quando lida normalmente ou de trás para a frente. Uma outra definição é dada por estas 4 regras:

- 1)  $\epsilon$  é um palíndromo.
- 2) Se  $a$  é um símbolo qualquer, então a cadeia ' $a$ ' é um palíndromo.
- 3) Se  $a$  é um símbolo qualquer e ' $x$ ' é uma cadeia, então a cadeia ' $axa$ ' é um palíndromo.
- 4) A única forma de se obter um palíndromo é aplicando, um número finito de vezes, as regras 1 a 3 acima descritas.

Prove, por indução, que as duas definições são equivalentes.

**Solução:** Seja  $\Sigma$  um alfabeto finito e  $a \in \Sigma$  um caracter qualquer.

**Demonstração.** Por indução em  $n$ , onde  $n \in \mathbb{N}$  é o número de caracteres  $a$  usados para formar um palíndromo.

*Base:* De fato, quando  $n = 0$ , obtemos uma cadeia nula a qual resulta no mesmo valor (a cadeia nula) quando lida normalmente ou de trás para frente. Do mesmo modo, com  $n = 0$  caracteres, obtemos  $\epsilon$  que, pela regra 1 acima, é um palíndromo. Logo, as definições são iguais para  $n = 0$ .

Analogamente, quando quando  $n = 1$ , obtemos uma cadeia formada por um único caracter  $a$ , que portanto pode ser lida normalmente ou de trás para frente. Do mesmo modo, com  $n = 1$  caracter, obtemos a cadeia ' $a$ ' que, pela regra 2 acima, é um palíndromo. Logo, as definições são iguais para  $n = 1$ .

*Hipótese Indutiva:* Seja ' $x$ ' um palíndromo formado por  $n$  caracteres. Podemos considerar  $n > 1$  pois já analisamos os casos em que  $n \leq 1$ . Quando adicionamos mais um caracter, obtemos a cadeia ' $axa$ ' que resulta no mesmo valor quando lida normalmente ou de trás para frente. Da mesma forma, pelas regras 3 e 4 acima, ' $axa$ ' é um palíndromo. Portanto, quando supomos que as definições são iguais para um  $n$  qualquer, vemos que são válidas para  $n + 1$ .

Assim, pelo PIF, ambas as definições são válidas para todo  $n \in \mathbb{N}$ .

□

**Exercício 1.6.** Mostre que as relações abaixo são relações de equivalência e defina suas classes de equivalência:

- a) A relação  $R_1$  nos inteiros onde  $iR_1j$  se e somente se  $i = j$ .
- b) A relação  $R_2$  das pessoas onde  $pR_2q$  se e somente se  $p$  e  $q$  nasceram na mesma hora do mesmo dia de um ano qualquer.
- c) A relação  $R_3$  é a mesma relação definida em (b) considerando “do mesmo ano” ao invés de “num ano qualquer”.

### Solução:

Para provar que  $R$  é uma relação de equivalência, basta mostrar que  $R$  possui as propriedades reflexiva, simétrica e transitiva.

- a) *Reflexiva*: Para todo  $i \in \mathbb{Z}$ , temos  $i = i$ . *Simétrica*: Para quaisquer  $i, j$  inteiros, temos:  $i = j \Leftrightarrow j = i$ . *Transitiva*: Para quaisquer  $i, j, k$  inteiros, temos:  $i = j \wedge j = k \Rightarrow i = k$ .

As classes de equivalência de  $R_1$  podem ser definidas por:

$$[i]_{R_1} = \{j \in \mathbb{Z} | j = i\}$$

Ou seja, a relação  $R_1$  particiona  $\mathbb{Z}$  em um número infinito enumerável de classes de equivalência.

- b) Considerando que  $p, q$  e  $r$  são três pessoas quaisquer. *Reflexiva*: Toda pessoa nasce no mesmo dia e hora que ela mesma. *Simétrica*: Se  $p$  nasceu no mesmo dia e hora que  $r$ , então  $r$  nasceu no mesmo dia e hora que  $p$ . *Transitiva*: Se  $p$  nasceu no mesmo dia e hora de  $q$ , o qual nasceu no mesmo dia e hora de  $r$  então  $p$  e  $r$  nasceram no mesmo dia e hora.

As classes de equivalência de  $R_2$  podem ser definidas por:

$$[q]_{R_2} = \{p \text{ é uma pessoa} \mid p \text{ nasceu no mesmo dia e hora de } q\}$$

Note-se, no entanto, que há  $366 \times 24 = 8784$  combinações de dias e horas num ano qualquer.

Portanto, a relação  $R_2$  particiona o conjunto das pessoas em 8784 classes, ou seja, num número *finito* de equivalência disjuntas.

- c) Valem as mesmas considerações vistas para  $R_2$  apenas trocando-se as expressões “nasceu no mesmo dia e hora” por “nasceu no mesmo dia, hora e ano”.

As classes de equivalência de  $R_3$  podem ser definidas por:

$$[q]_{R_3} = \{p \text{ é uma pessoa} \mid p \text{ nasceu no mesmo dia, hora e ano de } q\}$$

Como há um conjunto infinito enumerável de anos a relação  $R_3$  particiona o conjunto  $P$  de pessoas em um número infinito enumerável de classes de equivalência disjuntas.

**Exercício 1.8.** Prove que qualquer relação de equivalência  $R$  num conjunto  $S$  o particiona em classes de equivalência disjuntas.

### Solução:

*Definição 1:* Diz-se que um conjunto  $S$  é particionado por  $n$  subconjuntos  $S_i \subseteq S$  (com  $n \in \mathbb{N}^*$ ) quando:

$$i) \quad S = S_1 \cup S_2 \cdots \cup S_n = \bigcup_{i=1}^n S_i.$$

$$\text{ii) } S_i \cap S_j = \emptyset \quad \forall i, j \in \mathbb{N}^* \wedge i \leq n \wedge j \leq n \wedge i \neq j.$$

$$\text{iii) } S_i \neq \emptyset \quad \forall i \in \mathbb{N}^* \wedge i \leq n.$$

*Definição 2:* Diz-se que um conjunto  $S$  é particionado por um número infinito enumerável de subconjuntos  $S_i \subseteq S$  quando:

$$\text{i) } S = S_1 \cup S_2 \cdots = \bigcup_{i=1}^{\infty} S_i.$$

$$\text{ii) } S_i \cap S_j = \emptyset \quad \forall i, j \in \mathbb{N}^* \wedge i \neq j.$$

$$\text{iii) } S_i \neq \emptyset \quad \forall i \in \mathbb{N}^*.$$

*Definição 3:* Se  $R$  é uma relação de equivalência em  $S$  e  $x \in S$ , então as classes de equivalência com respeito a  $R$  no conjunto  $S$  são:

$$[x]_R = \{y \in S | yRx\}$$

*Definição 4:* O conjunto de todas as classes de equivalência de  $S$  é denominado  $S$  módulo  $R$  e denotado por:

$$S/R = \{[x]_R | x \in S\}$$

**Teorema:** Se  $R$  é uma relação de equivalência em  $S$  então  $S/R = \{[x]_R | x \in S\}$  é uma partição de  $S$ .

**Demonstração.** Vamos provar que:

a) A união de todas as classes de equivalência  $[x]_R$  é o conjunto  $S$ , isto é:

$$\bigcup_{x \in S} [x]_R = S$$

**Demonstração.** Por construção.

De fato, todo elemento de  $S$  pertence à sua própria classe de equivalência pois  $[x]_R = \{y \in S | yRx\}$  e, pela propriedade reflexiva em  $R$ ,  $xRx$ . Logo:

$$x \in S \Rightarrow x \in [x]_R \tag{1}$$

Portanto, se percorrermos todos os elementos de  $S$  e formos unindo suas respectivas classes de equivalência, iremos obter o próprio conjunto  $S$ :

$$x \in [x]_R \Rightarrow \bigcup_{x \in S} [x]_R = S$$

□

Atendemos assim ao item (i) da definição 2 acima.

b) As classes de equivalência são disjuntas. Ou seja, para todo  $x \in S$  e  $y \in S$  temos

$$x \neq y \Rightarrow [x]_R \cap [y]_R = \emptyset \tag{2}$$

**Demonstração.** Por contradição.

Supondo, por absurdo, que  $x \neq y \Rightarrow [x]_R \cap [y]_R \neq \emptyset$ , temos, da definição de intersecção e de conjunto vazio, que:

$$[x]_R \cap [y]_R \neq \emptyset \Rightarrow \exists z \in [x]_R \cap [y]_R \quad (3)$$

Mas, pela definição de intersecção de conjuntos:

$$z \in [x]_R \cap [y]_R \Rightarrow z \in [x]_R \wedge z \in [y]_R \quad (4)$$

Pela definição de classes de equivalência:

$$z \in [x]_R \Rightarrow zRx \quad (5)$$

$$z \in [y]_R \Rightarrow zRy \quad (6)$$

Da eq. 5 e por simetria em  $R$ :

$$zRx \Rightarrow xRz \quad (7)$$

Das equações 7 e 6 e por transitividade em  $R$ :

$$xRz \wedge zRy \Rightarrow xRy \quad (8)$$

Da eq. 8 e da definição de classe de equivalência temos:

$$xRy \Rightarrow x \in [y]_R \quad (9)$$

Seja  $a \in S$  um caracter qualquer tal que  $a \in [x]_R$ .

Por definição:

$$a \in [x]_R \Rightarrow aRx \quad (10)$$

Da eq. 10 e por simetria:

$$aRx \Rightarrow xRa \quad (11)$$

Aplicando a eq. 11 na eq. 8:

$$xRa \Rightarrow aRy \quad (12)$$

Portanto:  $a \in [y]_R$ .

Mas se  $a \in [x]_R \Rightarrow a \in [y]_R$ , então:

$$[x]_R \subseteq [y]_R \quad (13)$$

Formalmente, o raciocínio das equações 5 a 13 pode ser resumido como:

$$a \in [x]_R \Rightarrow a \in [y]_R \Leftrightarrow [x]_R \subseteq [y]_R \quad (14)$$

Analogamente, por simetria em  $R$ :  $zRy \Rightarrow zRx$ . Por transitividade em  $R$ :  $yRz \wedge zRx \Rightarrow yRx$ . Se  $b \in [y]_R$  então  $bRy$ . Pela propriedade transitiva e por  $yRx$  temos  $bRy$ . Portanto,  $b \in [x]_R$ . A relação  $b \in [y]_R \Rightarrow b \in [x]_R$  equivale a

$$[y]_R \subseteq [x]_R \quad (15)$$

Das equações 13 e 15 e da definição de igualdade de conjuntos, temos:

$$[x]_R \subseteq [y]_R \wedge [y]_R \subseteq [x]_R \Leftrightarrow [x]_R = [y]_R \quad (16)$$

Mas, se  $[x]_R = [y]_R$  então  $[x]_R \cap [y]_R \neq \emptyset$  o que é uma contradição com nossa hipótese inicial!

Assim, concluímos que, se  $x \in S$  e  $y \in S$ , então:

$$x \neq y \Rightarrow [x]_R = [y]_R \vee [x]_R \cap [y]_R = \emptyset \quad (17)$$

□

Atendemos assim ao item (ii) da definição 2 acima.

c) Nenhuma das classes de equivalência é um conjunto vazio.

**Demonstração.** Por construção.

Cada elemento em  $S$  pertence à sua própria classe de equivalência:

$$x \in S \Rightarrow x \in [x]_R \quad (18)$$

Portanto, cada classe de equivalência possui pelo menos um elemento.

□

Atendemos assim ao item (iii) da definição 2 acima, completando a demonstração do teorema proposto.

□

**Exercício 2.1.** Encontre um autômato cujo comportamento corresponda ao do circuito da figura abaixo, considerando que os estados finais correspondem a uma saída 1. Um círculo com um ponto representa uma porta-E cuja saída é 1 apenas quando ambas as entradas valem 1. Um círculo com um + representa uma porta-OU cuja saída é 1 sempre que pelo menos uma de suas entradas seja 1. Um círculo com um ~ representa um inversor, cuja saída é 1 quando a entrada é 0 e 0 quando a entrada é 1. Assuma que há tempo suficiente para que as mudanças na entrada se propaguem pelo circuito e a rede se estabilize.

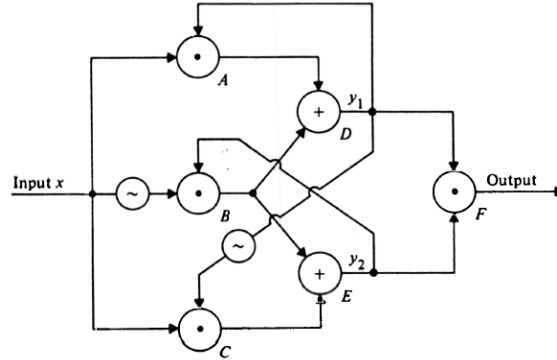


Figura 1 Circuito Lógico do Exercício 2.1.

**Solução:**

Este circuito possui realimentações:

- i) O sinal de  $y_1$  é enviado diretamente para a porta  $A$  e via inversor para a porta  $C$ .
- ii) O sinal de  $y_2$  é enviado diretamente para a porta  $B$ .

Podemos arbitrar valores para  $y_1$  e  $y_2$ , calculando o valor de todas as portas, o que irá resultar em novos valores para  $y_1$  e  $y_2$ . Com estes novos valores, recalculamos o valor de todas as portas, repetindo o processo até que os valores de  $y_1$  e  $y_2$  se estabilizem. Quando isso ocorre, obtemos o valor final da saída da porta  $F$ .

Criamos um software para simular as portas  $A$  a  $F$ , tal como definidas no enunciado, face os diferentes valores de  $y_1$  e  $y_2$ .

Nas tabelas abaixo, E1 e E2 representam as entradas e S a saída de cada porta. Quando as saídas das portas D e E são diferentes dos valores iniciais de  $y_1$  e  $y_2$ , respectivamente, acrescentamos uma nova linha à tabela e recalculamos os valores em cada porta.

**Parte 1:** Considerando  $x = 0$ .

Para  $y_1 = y_2 = 0$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Para  $y_1 = 0$  e  $y_2 = 1$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
0	0	1	0	0	0	1	1	1	0	1	0	0	1	1	1	0	1	1	1	1
0	1	1	0	1	0	1	1	1	0	0	0	0	1	1	1	0	1	1	1	1

Para  $y_1 = 1$  e  $y_2 = 0$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Para  $y_1 = y_2 = 1$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
0	1	1	0	1	0	1	1	1	0	0	0	0	1	1	1	0	1	1	1	1

**Parte 2:** Considerando  $x = 1$ .

Para  $y_1 = y_2 = 0$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
1	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0	1	0
1	0	1	1	0	0	0	1	0	1	1	1	0	0	0	0	1	1	0	1	0

Para  $y_1 = 0$  e  $y_2 = 1$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
1	0	1	1	0	0	0	1	0	1	1	1	0	0	0	0	1	1	0	1	0

Para  $y_1 = 1$  e  $y_2 = 0$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
1	1	0	1	1	1	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0

Para  $y_1 = y_2 = 1$  temos:

Variáveis			A			B			C			D			E			F		
$x$	$y_1$	$y_2$	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S	E1	E2	S
1	1	1	1	1	1	0	1	0	1	0	0	1	0	1	0	0	0	1	0	0
1	1	0	1	1	1	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0

**Parte 3:** Resumindo as 8 tabelas acima, temos:

$x$	Início		Fim		F
	$y_1$	$y_2$	$y_1$	$y_2$	
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	0	0

### Conclusão:

Agora podemos associar estados aos valores de  $y_1$  e  $y_2$ . Por exemplo: quando  $y_1 = 0$  e  $y_2 = 1$  temos o estado 01. Deste modo, analisando a segunda linha da tabela acima, vemos que quando  $x = 0$  o circuito vai do estado 01 para o estado 00. Usando raciocínio análogo para as outras linhas da tabela, definimos a seguinte função de transição:

→	Estado	0	1
	00	00	01
	01	11	01
	10	00	10
	11	11	10

O estado 11 é o estado final pois é o único em que  $F = 1$ .

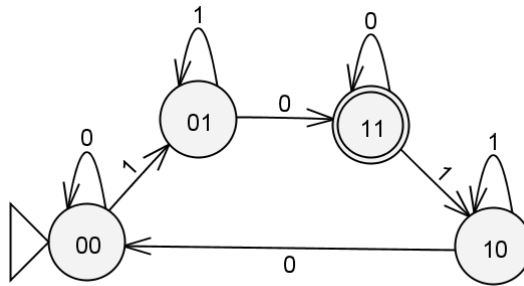


Figura 2 Autômata Finito do Exercício 2.1.

**Exercício 2.2.** Historicamente, autômatos finitos foram utilizados pela primeira vez para modelar redes neurais. Encontre o autômata finito cujo comportamento é equivalente ao da rede neural da Figura 3. Os estados finais do autômato correspondem a uma saída 1 na rede. Cada neurônio possui sinapses excitatórias (círculos brancos) e inibitórias (círculos pretos). Um neurônio produz uma saída 1 quando o número de sinapses excitatórias com entrada-1 é maior que o número de sinapses inibitórias com entrada-1 em quantidade maior ou igual ao limiar de excitação do neurônio (o número que aparece dentro do triângulo). Assuma que há tempo suficiente para que as mudanças na entrada se propaguem pelo neurônio e que a rede se estabilize. Assuma também que, inicialmente,  $y_1 = y_2 = y_3 = 0$ .

### Solução:

Nas tabelas abaixo, 1, 2 e 3 representam as entradas de cada neurônio. As letras  $E$  e  $I$  representam entradas excitadoras e inibidoras, respectivamente. A letra  $S$  representa a saída do neurônio.

Consideramos todos os valores  $y_1, y_2$  e  $y_3$ . Quando, a partir do valor inicial, calcula-se valores para  $y_1, y_2$  e  $y_3$  diferentes dos iniciais, cria-se uma nova linha na tabela, repetindo-se o cálculo destas variáveis até que o neurônio se estabilize.



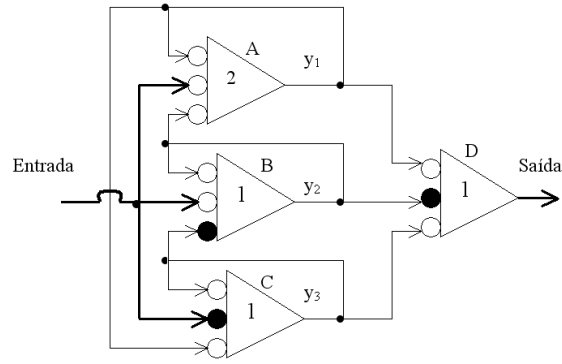


Figura 3 Rede Neural do Exercício 2.2.

Considerando  $x = 0$ .

Entrada			A				B				C				D			
$y_1$	$y_2$	$y_3$	1E	2E	3E	S	1E	2E	3I	S	1E	2I	3E	S	1E	2I	3E	S
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1
0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0
0	1	1	0	0	1	0	1	0	1	0	1	0	0	1	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1
1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1
1	0	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1
1	1	0	1	0	1	1	1	0	0	1	0	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	1
1	0	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1
1	1	1	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	1
1	0	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1

Considerando  $x = 1$ .

Entrada			A				B				C				D			
$y_1$	$y_2$	$y_3$	1E	2E	3E	S	1E	2E	3I	S	1E	2I	3E	S	1E	2I	3E	S
0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0
0	1	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0
0	0	1	0	1	0	0	0	1	1	0	1	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0
0	1	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0
0	1	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0
0	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0
1	0	0	1	1	0	1	0	1	0	1	0	1	1	0	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0
1	0	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

A partir das duas tabelas acima, e considerando cada estado como sendo formado pela concatenação dos valores de  $y_1, y_2$  e  $y_3$ , temos a seguinte função de transição:

Estado	0	1
→ 000	000	110
★ 001	001	110
010	010	110
011	001	110
100	001	110
★ 101	001	101
110	001	110
★ 111	001	111

Os estados 001, 101 e 111 são terminais pois resultam em saída igual a 1.

A partir da função de transição, vemos que apenas os estados 110 e 001 são alcançáveis a partir do estado inicial 000. Portanto, o autômata procurado é o mostrado na Figura 4.

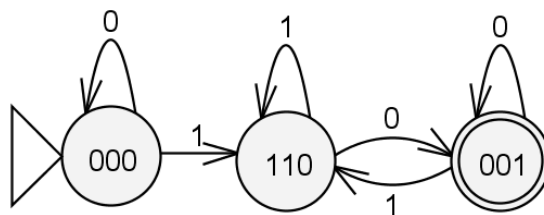


Figura 4 Autômato do Exercício 2.2.

---

**Exercício 2.3.** Considere o brinquedo mostrado na Figura 5. Uma bola é jogada em A ou em B. Alavancas  $x_1, x_2$  e  $x_3$  desviam a bola para a esquerda ou para a direita. Sempre que a bola se choca com uma alavanca, esta muda de posição de modo que a próxima bola seja levado para o lado oposto.

- a) Modele este brinquedo como um autômata finito. Considere que a bola caindo em A seja uma entrada 0 e em B uma entrada 1. Uma sequência de entradas é aceita quando a última bola desta sequência sair pela saída D.

- b) Descreva o conjunto das seqüências aceitas pelo automato.
- c) Modele o brinquedo como uma Máquina de Mealy cuja saída é a seqüência de  $C$ 's e  $D$ 's equivalente às posições nas quais as bolas saem.

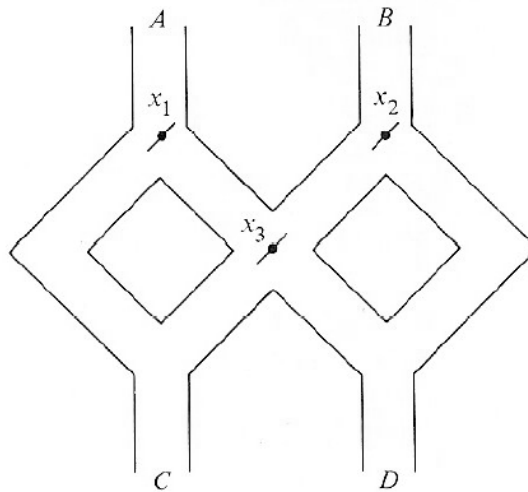


Figura 5 Brinquedo do Exercício 2.3.

### Solução:

- a) Em nosso autômata finito, as posições de  $x_1, x_2$  e  $x_3$ , tal como aparecem na Figura 5, são representadas por 0. Após terem sido atingidas por uma bola, a variável  $x_i$  muda para a posição oposta, isto é, se  $x_i = 0$  então a nova posição será  $x_i = 1$  e vice-versa. Também vamos considerar que a saída da bola por  $D$  equivale a uma saída 1 (estado final) e a saída por  $C$  equivale a um 0.

Esta tabela resume os diversos estados possíveis:

$x$	<i>Início</i>			<i>Fim</i>			Saída
	$x_1$	$x_2$	$x_3$	$x_1$	$x_2$	$x_3$	
0	0	0	0	1	0	0	0
0	0	0	1	1	0	1	0
0	0	1	0	1	1	0	0
0	0	1	1	1	1	1	0
0	1	0	0	0	0	1	0
0	1	0	1	0	0	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	0	1
1	0	0	0	0	1	1	0
1	0	0	1	0	1	0	1
1	0	1	0	0	0	0	1
1	0	1	1	0	0	1	1
1	1	0	0	1	1	1	0
1	1	0	1	1	1	0	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	1	1

A partir da tabela acima criamos um autômato finito determinista no qual cada estado é a concatenação dos valores de  $x_1, x_2, x_3$  e  $S$ . Como nosso alfabeto  $\Sigma = \{0, 1\}$  possui 2 caracteres, cada  $x_i$  e  $S$  podem

assumir 2 possíveis valores. Portanto, temos  $4^2 = 16$  possíveis estados. Sendo 0000 o estado inicial, todos os estados em que  $S = 1$  são estados de aceitação. Portanto:

		0	1
→	0000	1000	0110
★	0001	1000	0110
	0010	1010	0101
★	0011	1010	0101
	0100	1100	0001
★	0101	1100	0001
	0110	1110	0011
★	0111	1110	0011
	1000	0010	1110
★	1001	0010	1110
	1010	0001	1101
★	1011	0001	1101
	1100	0110	1001
★	1101	0110	1001
	1110	0101	1011
★	1111	0101	1011

Analisando a tabela acima, notamos que os estados 0100, 0111 e 1111 não são alcançados a partir do estado inicial. Portanto, o autônomo finito mínimo possui apenas 13 estados:

		0	1
→	0000	1000	0110
★	0001	1000	0110
	0010	1010	0101
★	0011	1010	0101
★	0101	1100	0001
	0110	1110	0011
	1000	0010	1110
★	1001	0010	1110
	1010	0001	1101
★	1011	0001	1101
	1100	0110	1001
★	1101	0110	1001
	1110	0101	1011

- b) Este autômata finito possui 13 estados e, portanto, a obtenção de uma expressão regular usando o algoritmo dado em aula é muito trabalhosa.

Optamos por analisar o funcionamento do brinquedo chegando a uma descrição em português para a linguagem reconhecida.

### Introdução

Inicialmente,  $x_2 = 0$ . Portanto, a bola que cai por  $B$  será desviada para  $x_3$ . No entanto, a próxima bola que cair em  $B$  será levada para  $D$  que é um estado de aceitação. Ou seja, as bolas de ordem ímpar (primeira, terceira, quinta etc) são levadas para  $x_3$  de onde podem ser aceitas ou não. Porém, as bolas de ordem par (segunda, quarta, sexta etc) que entram por  $B$  são levadas para um estado de aceitação, não importando os valores de  $x_1$  e  $x_3$ .

Portanto, se a última bola entrar por  $B$  e for de ordem par, a cadeia será aceita. Ou seja, o autômata finito reconhece cadeias que satisfazem a estas condições:

- i) Possuem um número par de 1's.
- ii) Terminam em 1.

Analogamente, bolas de ordem ímpar passando por  $A$  são rejeitadas, não importando os valores de  $x_2$  ou  $x_3$ .

Portanto, falta analisar o que acontece quando as bolas de ordem par que entraram por  $A$  e as de ordem ímpar que entram por  $B$  chegam em  $x_3$ .

### **Bolas que entram por B**

Se a bola entrou em  $B$  e chegou em  $x_3$  e é a última bola, então a cadeia de entrada tem um número ímpar de 1's.

Inicialmente,  $x_3 = 0$ . Portanto, se chegar um número par de bolas em  $x_3$  ele continuará zero e a última bola sairá por  $C$ . Portanto, para que a cadeia seja aceita, ela terá que enviar bolas em um número ímpar de vezes para  $x_3$ .

Se  $n_0$  é o número de bolas que entram por  $A$  (que é igual ao número de zeros na cadeia de entrada) então  $\lfloor \frac{n_0}{2} \rfloor$  bolas serão enviadas para  $x_3$ . Por exemplo, se 3 bolas entrarem por  $A$ , apenas  $\lfloor \frac{3}{2} \rfloor = 1$  será enviada para  $x_3$ .

Analogamente, se  $n_1$  é o número de bolas que entram por  $B$  (que é igual ao número de uns na cadeia de entrada) então  $\lceil \frac{n_1}{2} \rceil$  bolas serão enviadas para  $x_3$ . Por exemplo, se 3 bolas entrarem por  $B$ ,  $\lceil \frac{3}{2} \rceil = 2$  serão enviadas para  $x_3$ .

Portanto, o autômata finito reconhece cadeias que atendem simultaneamente a todas estas condições:

- i) Terminam em 1.
- ii) Têm um número ímpar de 1's.
- iii)  $\lfloor \frac{n_0}{2} \rfloor + \lceil \frac{n_1}{2} \rceil$  é ímpar.

### **Bolas que entram por A**

Analogamente, considerando que a bola entrou em  $A$ , definimos que o automato finito reconhece cadeias que atendem simultaneamente a todas estas condições:

- i) Terminam em 0 .
- ii) Têm um número par de 0's.
- iii)  $\lfloor \frac{n_0}{2} \rfloor + \lceil \frac{n_1}{2} \rceil$  é par.

### **Conclusão**

Se  $L$  é a linguagem do AF e  $x \in L$  então  $x$  é uma cadeia não-nula que satisfaz a uma destas 3 regras:

- i) Possui número par de 1's e termina em 1.
  - ii) Possui número ímpar de 1's, termina em 1 e  $\lfloor \frac{n_0}{2} \rfloor + \lceil \frac{n_1}{2} \rceil$  é ímpar.
  - iii) Possui número par de 0's, termina em 0 e  $\lfloor \frac{n_0}{2} \rfloor + \lceil \frac{n_1}{2} \rceil$  é par.
- c) O último dígito de cada estado de nosso autômato é o valor da saída. Portanto, a Máquina de Mealy contém os mesmos estados do AFD acima, porém imprimindo um caracter que é igual ao último dígito do estado para onde o AFD transita. Por exemplo, se estamos no estado 0000 e lemos um 0, transitamos para o estado 1000. Portanto, imprimimos o último dígito de 1000 que é zero.

Portanto a Máquina de Mealy é:

		Entrada		Saída	
		0	1	0	1
→	0000	1000	0110	0	0
★	0001	1000	0110	0	0
	0010	1010	0101	0	1
★	0011	1010	0101	0	1
★	0101	1100	0001	0	1
	0110	1110	0011	0	1
	1000	0010	1110	0	0
★	1001	0010	1110	0	0
	1010	0001	1101	1	1
★	1011	0001	1101	1	1
	1100	0110	1001	0	1
★	1101	0110	1001	0	1
	1110	0101	1011	1	1

**Exercício 2.4.** Suponha que  $\hat{\delta}$  é a função de transição estendida de um AFD. Prove que, para quaisquer cadeias  $x$  e  $y$ , temos:

$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$$

Dica: use indução em  $|y|$ .

**Solução:**

A função  $\hat{\delta}$  é definida recursivamente a partir da função de transição  $\delta$ :

$$\hat{\delta}(q, w) = \begin{cases} q & \text{se } w = \epsilon \\ \delta(\hat{\delta}(q, x), a) & \text{se } w = xa \text{ para } a \in \Sigma \text{ e } x \in \Sigma^+ \end{cases} \quad (19)$$

Informalmente, ambas as funções  $\delta$  e  $\hat{\delta}$  definem um novo estado a partir de um estado atual. Porém, a função  $\delta$  calcula o novo estado a partir de um único caracter enquanto que a função  $\hat{\delta}$  aceita uma cadeia completa como argumento. Neste sentido, a função  $\hat{\delta}$  “estende” a definição de  $\delta$ .

**Demonstração.** Vamos provar, por indução em  $n = |y|$ , que para quaisquer  $x$  e  $y$  em  $\Sigma^*$ :

$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y) \quad (20)$$

*Base:* Fazendo  $n = 0$  temos:

$$n = 0 \Rightarrow |y| = 0 \Rightarrow y = \epsilon \Rightarrow xy = x \quad (21)$$

Se  $xy = x$  então, por definição, o lado esquerdo da eq. 20 pode ser escrito como:

$$\hat{\delta}(q, xy) = \hat{\delta}(q, x) \quad (22)$$

Analogamente, aplicando  $y = \epsilon$  no lado direito da eq. 20, temos:

$$\hat{\delta}(\hat{\delta}(q, x), y) = \hat{\delta}(\hat{\delta}(q, x), \epsilon) \quad (23)$$

Aplicando a eq. 19 (caso  $w = \epsilon$ ) na equação acima, temos:

$$\hat{\delta}(\hat{\delta}(q, x), \epsilon) = \hat{\delta}(q, x) \quad (24)$$

Das equações 23 e 24, temos:

$$\hat{\delta}(\hat{\delta}(q, x), y) = \hat{\delta}(q, x) \quad (25)$$

Portanto, pelas equações 22 e 25, temos que a eq. 20 é válida para  $n = 0$ .

*Hipótese Indutiva:* Supondo que a eq. 20 é válida para  $n - 1$  vamos provar que ela também é válida para  $n$ .

De fato, fazendo  $y = za$  para um certo  $a \in \Sigma$  temos que:

$$|y| = n \Rightarrow |za| = |n| \Rightarrow |z| + |a| = n \Rightarrow |z| + 1 = n \Rightarrow |z| = n - 1 \quad (26)$$

Se  $y = za$  então o lado direito da eq. 20 pode ser reescrito como:

$$\hat{\delta}(\hat{\delta}(q, x), y) = \hat{\delta}(\hat{\delta}(q, x), za) \quad (27)$$

Apenas para facilitar a notação, fazemos  $\hat{\delta}(q, x) = p$  para um certo  $p \in Q$ . Portanto, o lado direito da equação acima pode ser reescrito como:

$$\hat{\delta}(\hat{\delta}(q, x), za) = \hat{\delta}(p, za) \quad (28)$$

Mas, pela eq. 19 (caso  $w = xa$ ), o lado direito da equação acima pode ser reescrito como:

$$\hat{\delta}(p, za) = \delta(\hat{\delta}(p, z), a) \quad (29)$$

Substituindo  $\hat{\delta}(q, x) = p$ , podemos reescrever o lado direito da equação acima como:

$$\delta(\hat{\delta}(p, za), a) = \delta(\hat{\delta}(\hat{\delta}(q, x), z), a) \quad (30)$$

Mas, pela Hipótese Indutiva, temos que  $\hat{\delta}(q, xz) = \hat{\delta}(\hat{\delta}(q, x), z)$ . Substituindo este resultado na equação acima, temos:

$$\delta(\hat{\delta}(\hat{\delta}(q, x), z), a) = \delta(\hat{\delta}(q, xz), a) \quad (31)$$

Da eq. 19, sabemos que  $\hat{\delta}(q, xza) = \delta(\hat{\delta}(q, xz), a)$ . Substituindo este resultado na equação acima, temos:

$$\delta(\hat{\delta}(q, xz), a) = \hat{\delta}(q, xza) \quad (32)$$

Substituindo  $y = za$  na equação acima temos:

$$\hat{\delta}(q, xza) = \hat{\delta}(q, xy) \quad (33)$$

Na eq. 27, começamos com o lado direito da eq. 20 e, com o uso da hipótese indutiva, chegamos à eq. 33 que é o lado esquerdo da eq. 20. Ou seja, supondo que a expressão é válida para  $n - 1$  mostramos que ela é válida para  $n$ .

Portanto, pelo Princípio da Indução Finita (PIF), a expressão  $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$  vale para todo  $n \in \mathbb{N}$ .  $\square$

**Exercício 2.5.** Encontre os autômatas finitos deterministas que aceitam as seguintes linguagens no alfabeto  $\Sigma = \{0, 1\}$ :

a) O conjunto de todas as cadeias terminando em 00.

- b) O conjunto de todas as cadeias com três 0's consecutivos.
- c) O conjunto de todas as cadeias tais que cada bloco de 5 símbolos consecutivos contenha pelo menos dois 0's.
- d) O conjunto de todas as cadeias começando por 1 tais que, interpretadas como a representação em binário de um número inteiro, resultam num valor congruente com zero módulo cinco.
- e) O conjunto de todas as cadeias tais que o décimo símbolo, contado do fim da cadeia em direção a seu início, seja igual a 1.

### Solução:

- a) Iniciamos no estado  $[-]$  no qual a cadeia não possui nenhum 0. Se encontrarmos um 0, vamos para o estado  $[0]$ . Neste estado, ao encontrarmos um novo 0 vamos para o estado  $[00]$  que é o estado de aceitação. De qualquer estado, ao encontrarmos um 1 retornamos ao estado inicial. Ficamos no estado final  $[00]$  apenas se continuarmos encontrando 0's na cadeia de entrada. Deste modo, garantimos estar neste estado apenas se o último carácter lido for um 0.

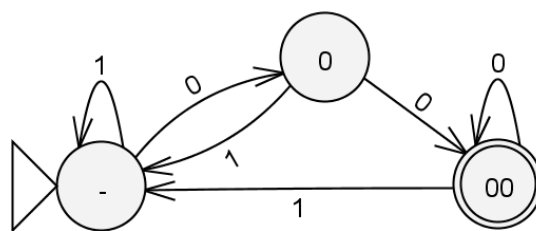


Figura 6 Diagramas de transição para o Exercício 2.5 (a).

- b) Este autômato é semelhante ao anterior. O estado final  $[000]$  é alcançado após terem sido encontrados três 0's consecutivos. Uma vez que tenhamos atingido o estado final, nele permanecemos quaisquer que sejam os demais caracteres lidos. Não precisamos sair deste estado pois não é necessário que os três caracteres 0 estejam no final da cadeia.

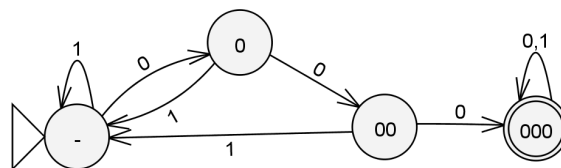


Figura 7 Diagramas de transição para o Exercício 2.5 (b).

- c) Em cada instante, analisamos 5 posições consecutivas. Cada posição pode ser ocupada por um dentre três símbolos:  $\epsilon$ , 0 ou 1. Portanto, este AF tem  $3^5 = 243$  estados!

São estados finais os que possuírem 0's em quaisquer duas posições nas cinco posições possíveis. Portanto, existem  $\binom{5}{2} = \frac{5 \cdot 4}{2} = 10$  estados finais.

Vamos definir cada estado pelos possíveis caracteres em cada posição. Vamos começar no estado inicial  $\epsilon$  e vamos adicionando um carácter até termos encontrado todos os 10 estados finais.

Vamos criar um estado "FIM" no qual o AF irá permanecer quando houver menos que dois zeros em cinco posições. Uma vez que o AF tenha atingido o estado "FIM", ele irá permanecer neste estado, não importando os demais caracteres lidos.



Deste modo, não será necessário mapear todos os 243 estados mas apenas aqueles que levam ao estado “FIM” ou a um dos 10 estados finais.

Outra forma de reduzir o número de estados do AF é ir para o estado 00 toda vez que forem encontrados dois zeros consecutivos.

Com estas considerações, obtemos este AF:

		0	1
→	ε	0	1
	0	00	01
	1	10	11
*	00	00	001
	01	010	011
	10	00	101
	11	110	111
*	001	010	0011
	010	00	0101
	011	0110	0111
	101	010	1011
	110	00	1101
	111	1110	FIM
*	0011	0110	00111
*	0101	010	01011
*	0110	00	01101
*	0111	01110	FIM
	1011	0110	FIM
	1101	010	FIM
	1110	00	FIM
*	00111	01110	FIM
*	01011	0110	FIM
*	01101	010	FIM
*	01110	00	FIM
	FIM	FIM	FIM

d) Seja  $w$  a string lida e  $x$  o número resultante quando  $w$  é convertida de binário para um número inteiro. Iremos aceitar  $w$  quando:

- O primeiro caracter de  $w$  for 1. Portanto, se o primeiro caracter for 0, o AF vai para um estado “FIM” onde permanece até o término da leitura da cadeia de entrada.
- O resto da divisão de  $x$  por 5 for igual a 0. Portanto, o AF deve ter estados 0, 1, 2, 3 e 4, cada qual representando o resto da divisão de  $x$  por 5. O estado de aceitação é o 0, pois é aquele em que o resto da divisão de  $x$  por 5 é zero e, portanto,  $x$  é congruente com zero módulo cinco.

Para obter a transição entre os estados 0, 1, 2, 3 e 4 analisamos alguns possíveis valores de  $w$

$w$	$x$	$x \bmod 5$
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	0
110	6	1
111	7	2
1000	8	3
1001	9	4
1010	10	0
1011	11	1
1100	12	2
1101	13	3
1110	14	4
1111	15	0

Com estas considerações, obtemos este AF (Figura 8):

		0	1
$\rightarrow$	$\epsilon$	FIM	1
	FIM	FIM	FIM
$\star$	0	0	1
	1	2	3
	2	4	0
	3	1	2
	4	3	4

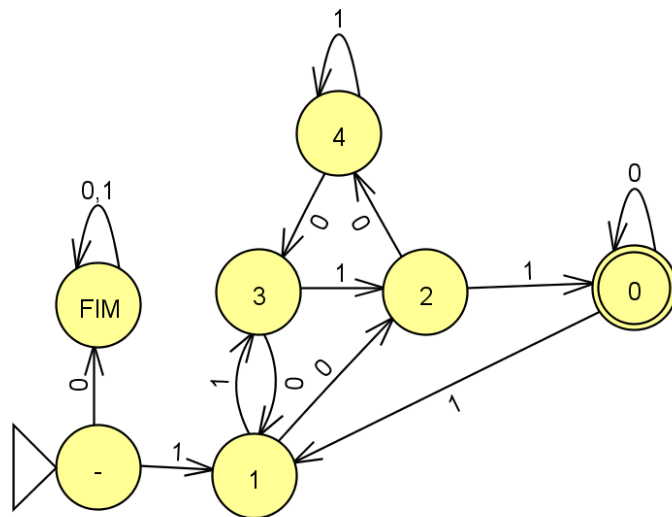


Figura 8 Diagramas de transição para o Exercício 2.5 (d).

e) Este autômato tem que monitorar os 10 últimos caracteres lidos.

Para monitorar apenas o último caracter lido teríamos que ter 3 estados. Um para monitorar a string nula  $\epsilon$  e mais dois para monitorar os possíveis valores 0 ou 1. Estes estados poderiam ser  $Q = \{\epsilon, 0, 1\}$ .

Analogamente, para monitorar os 2 últimos caracteres, teríamos que ter 7 estados. Um estado para  $\epsilon$ , mais 3 estados possíveis para o primeiro caracter (0 ou 1 ou  $\epsilon$ ) vezes 2 estados (0 ou 1) para o último

caracterer. Estes estados poderiam ser  $Q = \{\epsilon, 0, 1, 00, 01, 10, 11\}$ .

Continuando nosso raciocínio, para monitorar os 3 últimos caracteres, teríamos um estado para  $\epsilon$ , mais 3 estados possíveis para o primeiro caracter (0 ou 1 ou  $\epsilon$ ) vezes 3 estados para o segundo caracterer (0 ou 1 ou  $\epsilon$ ), vezes 2 estados para o último caracterer (0 ou 1) resultando em  $1 + 3 * 3 * 2 = 19$  estados.

Logo, para monitorar 10 estados, teríamos que ter  $1 + 3^9 * 2 = 39367$  estados! Portanto, ao invés de enumerar exaustivamente estes estados, iremos apenas identificá-los com expressões matemáticas que definem  $M = (Q, \Sigma, \delta, q_0, F)$ .

Consideramos que, para nosso autômato, temos  $\Sigma = \{0, 1\}$ .

Seja  $a_i \in \Sigma \cup \{\epsilon\}$  um caracter qualquer e  $n \in \mathbb{N}$ .

Definimos os possíveis estados do autômato:

$$Q = \{[a_1 a_2 \dots a_n] | 1 \leq n \leq 10\}$$

Ou seja,  $Q$  contém os estados  $\{\epsilon, 0, 1, 00, 01, 10, 11, \dots, 1111111111\}$  que correspondem aos possíveis 10 últimos caracteres lidos pelo autômata.

O conjunto dos estados de aceitação  $F \subset Q$  é formado pelos estados de  $Q$  que contém 10 caracteres  $a_i \in \Sigma$  e o primeiro destes caracteres é  $a_1 = 1$ . Formalmente:

$$F = \{[1a_2 \dots a_n] | \forall a_i \in \Sigma \wedge 2 \leq i \leq n \wedge n = 10\}$$

Considerando  $b \in \Sigma$  um caracter qualquer lido pelo autômato quando este está no estado  $[a_1 a_2 \dots a_n]$ , a função de transição  $\delta$  pode ser definida como:

$$\delta([a_1 a_2 \dots a_n], b) = \begin{cases} [a_1 a_2 \dots a_n b] & \text{se } n < 10 \\ [a_2 \dots a_n b] & \text{se } n = 10 \end{cases}$$

Por construção,  $0 \leq n \leq 10$  e não precisamos considerar os casos em que  $n > 10$ .

**Exercício 2.6.** *Descreva em português os conjuntos aceitos pelos autômatas finitos cujos diagramas de transição são dados na Figura 9.*

**Solução:**

- a) O conjunto das cadeias formadas por 0 ou mais caracteres 0 seguidos por 0 ou mais caracteres 1.
- b) O conjunto das cadeias que terminam em 101.
- c) O conjunto das cadeias que, começando com zero, possuem um número par de 0's e zero ou um número ímpar de 1's ou, começando com 1, tenha um número par de zeros e um número par de 1's. Ou seja, o primeiro caracter é a paridade da cadeia.

Como zero é par, o autômata aceita a cadeia nula (que tem zero caracteres 0 e zero caracteres 1).

**Exercício 2.7.** *Prove que o autômata finito, cujo diagrama de transição é dado na Figura 10, aceita o conjunto de cadeias sobre o alfabeto  $\Sigma = \{0, 1\}$  com igual número de 0's e 1's tal que cada prefixo tenha no máximo um 0 a mais que o número de 1's e no máximo um 1 a mais que o número de 0's.*

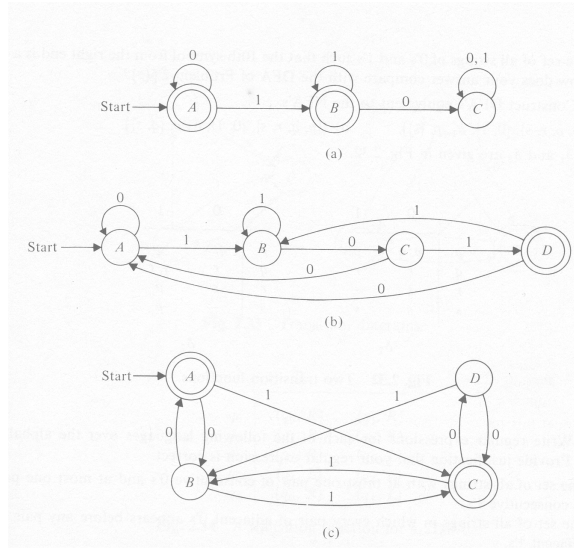


Figura 9 Diagramas de transição para o Exercício 2.6.

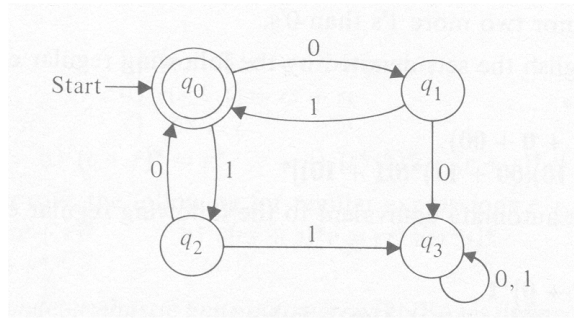


Figura 10 Diagrama de transição para o Exercício 2.7.

### Solução:

Seja  $a \in \Sigma$  e  $x \in \Sigma^*$ , respectivamente, um caracter e uma cadeia quaisquer.

Seja  $Q(a, x)$  uma função que devolve o número de vezes em que o caracter  $a$  é encontrado na cadeia  $x$ . Por exemplo:  $Q(0, 101) = 1$  e  $Q(1, 101) = 2$ .

*Definição 1:* Diz-se que  $x$  é uma cadeia “própria” quando, para todo prefixo  $p$  de  $x$  temos

$$|Q(0, p) - Q(1, p)| \leq 1$$

*Definição 2:* Diz-se que  $x$  é uma cadeia “balanceada” quando

$$Q(0, x) = Q(1, x)$$

Para o autômato da Figura 10 consideramos  $\delta$  sua função de transição e  $L$  a linguagem por ele aceita.

A partir destas considerações, temos que:

- i)  $\hat{\delta}(q_0, x) = q_0 \iff x$  é própria e balanceada. Portanto, este é o estado de aceitação.
- ii)  $\hat{\delta}(q_0, x) = q_1 \iff x$  é própria e  $Q(0, x) = Q(1, x) + 1$ .

iii)  $\hat{\delta}(q_0, x) = q_2 \iff x$  é própria e  $Q(0, x) = Q(1, x) - 1$ .

iv)  $\hat{\delta}(q_0, x) = q_3 \iff x$  não é própria.

Portanto, queremos provar que o automato da Figura 10, formalmente descrito acima, reconhece a linguagem  $L$  formada pelo conjunto de cadeias sobre o alfabeto  $\Sigma = \{0, 1\}$  com igual número de 0's e 1's tal que cada prefixo tenha no máximo um 0 a mais que o número de 1's e no máximo um 1 a mais que o número de 0's.

**Demonstração.** Vamos provar, por indução em  $n = |x|$ , que as considerações acima são válidas para qualquer  $x \in \Sigma^*$ .

*Base:* Se  $n = 0$  temos que  $|x| = 0$ . Por definição,  $x = \epsilon$ . Como  $q_0$  é o estado inicial do autômato e  $\hat{\delta}(q_0, \epsilon) = q_0$ , temos, pela condição (i) acima, que a cadeia vazia é aceita pelo autômato.

Como, de fato,  $\epsilon \in L$ , temos que as considerações são válidas para  $n = 0$ .

*Hipótese Indutiva:* Vamos supor, por hipótese, que temos uma cadeia  $x$  de comprimento  $|x| = n$ . Vamos mostrar que, ao adicionar a  $x$  um caracter  $a \in \Sigma$  formamos uma cadeia  $y = xa$  de comprimento  $|y| = |x| + 1$  e que  $y$  também está em um dos estados  $q_0, q_1, q_2$  ou  $q_3$  acima listados.

Analisando a hipótese indutiva para todos os estados possíveis, temos:

1) Supondo que  $\hat{\delta}(q_0, x) = q_0$ . Neste caso, pelo item (i) acima,  $x$  é própria e balanceada. Portanto:  $Q(0, x) = Q(1, x)$ .

Quando adicionamos o caracter  $a$  a  $x$  para obter  $y = xa$ , temos duas hipóteses.

Se  $a = 0$ , temos que  $Q(0, y) = Q(0, x) + 1 = Q(1, x) + 1 = Q(1, y) + 1$ . Portanto, pelo item (ii) acima:  $Q(0, y) = Q(1, y) + 1 \Rightarrow \hat{\delta}(q_0, y) = q_1$ .

Analogamente, se  $a = 1$ , temos que  $Q(1, y) = Q(1, x) + 1 = Q(0, x) + 1 = Q(0, y) + 1$ . Portanto, pelo item (iii) acima:  $Q(0, y) = Q(1, y) - 1 \Rightarrow \hat{\delta}(q_0, y) = q_2$ .

2) Supondo que  $\hat{\delta}(q_0, x) = q_1$ . Neste caso, pelo item (ii),  $x$  é própria e  $Q(0, x) = Q(1, x) + 1$ .

Quando fazemos  $y = xa$ , temos duas hipóteses.

Se  $a = 0$ , temos que  $Q(0, y) = Q(0, x) + 1 = Q(1, x) + 1 + 1 = Q(1, y) + 2$ . Portanto,  $|Q(0, y) - Q(1, y)| = 2 > 1$ . Portanto, pelo item (iv),  $y$  não é própria e  $\hat{\delta}(q_0, y) = q_3$ .

Analogamente, se  $a = 1$ , temos que  $Q(1, y) = Q(1, x) + 1 = Q(0, x) - 1 + 1 = Q(0, x) = Q(0, y)$ . Portanto,  $Q(0, y) = Q(1, y)$  e, pelo item (i),  $\hat{\delta}(q_0, y) = q_0$ .

3) Supondo que  $\hat{\delta}(q_0, x) = q_2$ . Neste caso, pelo item (ii),  $x$  é própria e  $Q(0, x) = Q(1, x) - 1$ .

Quando fazemos  $y = xa$ , temos duas hipóteses.

Se  $a = 0$ , temos que  $Q(0, y) = Q(0, x) + 1 = Q(1, x) - 1 + 1 = Q(1, x) = Q(1, y)$ . Portanto,  $Q(0, y) = Q(1, y)$ . Portanto, pelo item (i),  $\hat{\delta}(q_0, y) = q_0$ .

Analogamente, se  $a = 1$ , temos que  $Q(1, y) = Q(1, x) + 1 = Q(0, x) + 1 + 1 = Q(0, x) + 2 = Q(0, y) + 2$ . Portanto,  $|Q(0, y) - Q(1, y)| = 2 > 1$ . Portanto, pelo item (iv),  $y$  não é própria e  $\hat{\delta}(q_0, y) = q_3$ .

4) Supondo que  $\hat{\delta}(q_0, x) = q_3$ . Portanto, pelo item (iv),  $x$  não é própria. Quando fazemos  $y = xa$ , não importando o valor de  $a$  temos que  $y$  não é própria pois o fato de acrescentarmos sufixos não altera o prefixo da cadeia e apenas os sufixos são necessários para definir se uma cadeia é ou não é própria. Logo, pelo item (iv),  $\hat{\delta}(q_0, y) = q_3$ .

Portanto, pelo Princípio da Indução Finita, provamos que o AF aceita o conjunto de cadeias sobre o alfabeto  $\Sigma = \{0, 1\}$  com igual número de 0's e 1's tal que cada prefixo tenha no máximo um 0 a mais que o número de 1's e no máximo um 1 a mais que o número de 0's.

□

---

**Exercício 2.8.** Encontre os autômatos finitos não deterministas que aceitam as seguintes linguagens:

a) O conjunto das cadeias em  $(0 + 1)^*$  tal que dois caracteres 0 estejam separados por uma string de comprimento  $4i$  para algum  $i \geq 0$ .

b) O conjunto das cadeias sobre o alfabeto  $\Sigma = \{a, b, c\}$  que têm o mesmo valor, quando calculadas da direita para a esquerda e da esquerda para a direita, quando multiplicadas de acordo com a seguinte tabela:

	$a$	$b$	$c$
$a$	$a$	$a$	$c$
$b$	$c$	$a$	$b$
$c$	$b$	$c$	$a$

c) O conjunto das cadeias de 0's e 1's tal que o décimo caracter da direita para a esquerda é 1. Como este AF se compara com aquele da questão 2.5(e)?

**Solução:**

a) Procuramos definir um AFND que reconheça cadeias na forma  $w = \dots 0XX \dots XX0 \dots$  em que  $X$  pode valer 0 ou 1 e o comprimento da cadeia  $XX \dots XX$  é múltiplo de 4.

Para facilitar o raciocínio, vamos chamar a cadeia  $XX \dots XX$  de “cadeia intermediária”. Portanto, estamos interessados em reconhecer cadeias que possuem cadeias intermediárias de comprimento múltiplo de 4.

Uma primeira idéia de solução seria o AFD da Figura 11.

Neste AFD, os estados  $\{0\}$ ,  $\{1\}$ ,  $\{2\}$  e  $\{3\}$  medem o comprimento da cadeia entre dois 0's consecutivos. Ou seja, se o AFD está no estado  $\{1\}$  isto significa que o comprimento da cadeia intermediária, naquele instante, ao ser dividido por 4 deixa resto 1.

O estado  $\{FIM\}$  é o estado de aceitação.

Inicialmente, o AFD está no estado  $\{-\}$  de onde sai quando encontra o primeiro 0. Em seguida, se for encontrado o segundo 0, o AFD transita para o estado  $\{FIM\}$ . Este é o caso em que a cadeia intermediária é nula.

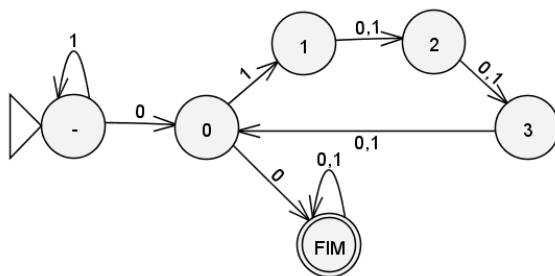


Figura 11 AFD que “tenta” resolver o Exercício 2.8(a).

Outro modo do AFD transitar para  $\{FIM\}$  é passando pelos estados  $\{1\}$ ,  $\{2\}$  e  $\{3\}$ . Ao retornar para o estado  $\{0\}$  garante-se que o comprimento da cadeia intermediária é múltiplo de 4. Neste instante, se for encontrado o segundo 0, o AFD transita para  $\{FIM\}$  e reconhece a cadeia.

Este AFD, tal como proposto, reconhece as cadeias 00 e 011110. Porém, ele não reconhece a cadeia 01011110 que pertence à linguagem. Este erro acontece porque, ao ler o segundo 0, o AFD está no estado  $\{2\}$  e não no estado  $\{0\}$ . Para resolver este problema, fazemos uma pequena alteração no AFD

da Figura 11 adicionando um não-determinismo no estado  $\{-\}$ . Deste modo, sempre que surge um 0 na cadeia, há uma transição do estado  $\{-\}$  para o estado  $\{0\}$  e o raciocínio acima descrito para o AFD se repete. Deste modo, cadeias como a 01011110 são corretamente reconhecidas.

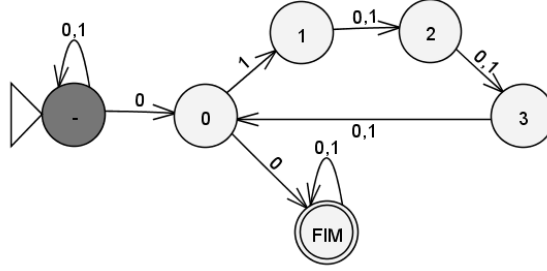


Figura 12 AFND que resolve o Exercício 2.8(a).

- b) Seja  $V(w)$  uma função que retorna o valor da cadeia  $w \in \Sigma^*$  multiplicando cada um de seus caracteres de acordo com a tabela dada no exercício.

Por exemplo, se  $w = cbca$  então:

$$V(w) = V(cbca) = ((c \times b) \times c) \times a = (c \times c) \times a = a \times a = a \quad (34)$$

Analogamente:

$$V(w^R) = V(acbc) = ((a \times c) \times b) \times c = (c \times b) \times c = c \times c = a \quad (35)$$

Portanto, a linguagem  $L$  pode definida como:

$$L = \{x \in \Sigma^* | V(x) = V(x^R)\}$$

Nossa estratégia é criar um AFD  $M_a$  que reconhece a linguagem:

$$L_a = \{x \in \Sigma^* | V(x) = a\}$$

Em seguida, criaremos um AFD  $M_a^R$  que reconhece a linguagem:

$$L_a^R = \{x \in \Sigma^* | V(x^R) = a\}$$

Analogamente, criamos os AFD's  $M_b, M_b^R, M_c$  e  $M_c^R$  que reconhecem, respectivamente, as linguagens  $L_b, L_b^R, L_c$  e  $L_c^R$ .

Como os possíveis valores de  $V(x)$  são apenas  $a, b$  e  $c$  temos que:

$$L = (L_a \cap L_a^R) \cup (L_b \cap L_b^R) \cup (L_c \cap L_c^R) \quad (36)$$

Se  $M$  reconhece  $L$  e  $M_i^\cap$  é um automata que reconhece  $(L_i \cap L_i^R)$  para  $i \in \{a, b, c\}$  então:

$$M = M_a^\cap \cup M_b^\cap \cup M_c^\cap \quad (37)$$

Agora nos resta apenas, demonstrar como se pode construir  $M$ .

**Construindo o AFD  $M_a$**

$M_a = (Q, \Sigma, \delta_a, q_0, F_a)$  onde:

- i)  $Q = \{-, A, B, C\}$ .
- ii)  $\Sigma = \{a, b, c\}$ .

iii)  $\delta_a$  é dada por:

		a	b	c
$\rightarrow$	-	A	B	C
$\star$	A	A	A	C
	B	C	A	B
	C	B	C	A

iv)  $q_0 = \{-\}$ .

v)  $F_a = \{A\}$ .

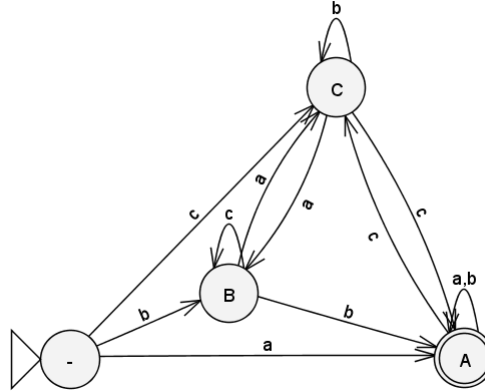


Figura 13 AFD  $M_a$  do Exercício 2.8(b).

Nota-se que  $\delta_a$  é um mimetismo da tabela de multiplicação dada. Por exemplo: na tabela de multiplicação,  $b \times a = c$  e, na função  $\delta_a$ , saindo do estado  $B$ , ao ler um  $a$ , o AFD transita para  $C$ .

Assim, se a cadeia  $x$  é formada pelos caracteres  $x_1x_2 \dots x_n$  então as transições do AFD  $M_a$  acima definido equivalem a ir multiplicando os valores  $x_i$  de  $x$  de forma análoga ao que é feito pela função  $V$ . A cada caracter lido, o estado de  $M_a$  indica o resultado parcial desta multiplicação até que, quando é lido o último caracter  $x_n$ , chega-se ao estado igual ao valor de  $V(x)$ . Compare, por exemplo, os resultados intermediários da equação 34 acima ( $c \times b = c$ ), ( $c \times c = a$ ) e ( $a \times a = a$ ) com as transições de  $M_a$  ao processar  $cbca$ :  $\{-\}, \{C\}, \{C\}, \{A\}, \{A\}$

Como o estado de aceitação de  $M_a$  é  $A$ , temos que este AFD irá aceitar apenas as cadeias  $x$  em que  $V(x) = a$ .

#### Construindo os AFD's $M_b$ e $M_c$

De forma análoga, apenas mudando o estado de aceitação para  $F_b = \{B\}$  e  $F_c = \{C\}$ , respectivamente, criamos os AFD's  $M_b = (Q, \Sigma, \delta_b, q_0, F_b)$  e  $M_c = (Q, \Sigma, \delta_c, q_0, F_c)$  que reconhecem as cadeias em que  $V(x) = b$  e  $V(x) = c$ , respectivamente.

As funções de transição  $\delta_b$  e  $\delta_c$  são as mesmas de  $\delta_a$  mostrada acima, com o mesmo estado inicial  $q_0 = \{-\}$ . As únicas diferenças são os estados de aceitação  $F_b = \{B\}$  e  $F_c = \{C\}$  já citados.

#### Construindo o AFND $M_a^R$

A partir do AFD  $M_a$  que reconhece as cadeias  $x$  em que  $V(x) = a$ , podemos criar o AFND  $M_a^R = (Q, \Sigma, \delta_a^R, q_0^R, F_a^R)$  que reconhece as cadeias  $x$  em que  $V(x^R) = a$ , fazendo as seguintes alterações em  $M_a$ :

- Inverter todas as transições de  $M_a$ . Por exemplo: em  $M_a$  existe uma transição do estado  $\{-\}$  para o estado  $\{A\}$  quando se lê o caracter  $a$ . Portanto, em  $M_a^R$  esta transição ocorrerá do estado  $\{A\}$  para o estado  $\{-\}$  quando se lê o caracter  $a$ .
- Inverter os estados inicial e final. Portanto,  $F_a^R = \{-\}$  e  $q_0^R = \{A\}$ .



Note-se que os estados de transição  $Q$  e o alfabeto  $\Sigma$  são iguais nos autômatos  $M_a$  e  $M_a^R$ .

No entanto,  $\delta_a^R$  é dada por:

		a	b	c
$\star$	-	$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow$	A	A	A	C
	B	C	-	B
	C	B	C	-

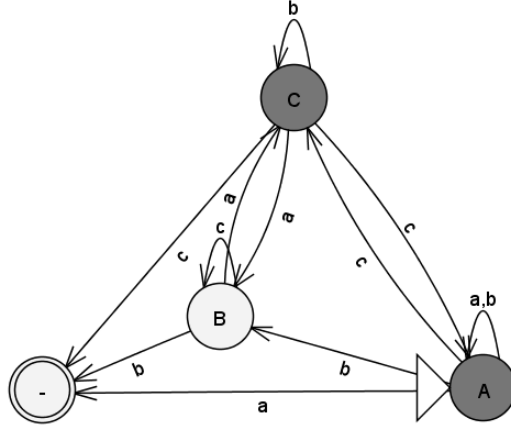


Figura 14 AFND  $M_a^R$  do Exercício 2.8(b).

#### Construindo os AFND's $M_b^R$ e $M_c^R$

A estratégia usada para obter  $M_a^R$  a partir de  $M_a$  pode ser usada, *mutatis mutandis*, para obter  $M_b^R$  a partir de  $M_b$  e  $M_c^R$  a partir de  $M_c$ .

#### Construindo o AFND $M_a^\cap = M_a \cap M_a^R$

Vamos construir  $M_a^\cap = (Q^\cap, \Sigma, \delta_a^\cap, q_0^\cap, F_a^\cap)$  a partir de  $M_a = (Q, \Sigma, \delta_a, q_0, F_a)$  e  $M_a^R = (Q, \Sigma, \delta_a^R, q_0^R, F_a^R)$ . Informalmente,  $M_a^\cap$  atua sobre um produto cartesiano dos estados de  $M_a$  e  $M_a^R$  tomando apenas os estados de  $M_a \cap M_a^R$ .

Assim, definimos:

- i)  $Q^\cap = Q \times Q$ . Para simplificar nossa notação iremos escrever  $q_{uv} = \{q_u, q_v\} \in Q^\cap$ .
- ii)  $\Sigma = \{a, b, c\}$ .
- iii)  $\delta_a^\cap$  é definida a partir de  $\delta_a$  e  $\delta_a^R$ , garantindo que  $M_a^\cap = M_a \cap M_a^R$ :

$$\delta_a^\cap(q_{uv}, w) = \{q_{xy} \mid q_x = \delta_a(q_u, w) \wedge q_y \in \delta_a^R(q_v, w)\} \quad (38)$$

- iv)  $q_0^\cap$  é o estado inicial de  $M_a^\cap$ . A partir de  $q_0^\cap$  criamos duas transições  $\epsilon$ : uma para o estado inicial de  $M_a$  e outra para o estado inicial de  $M_a^R$ .
- v)  $F_a^\cap$  é o estado final de  $M_a^\cap$ . Criamos duas transições  $\epsilon$ : uma do estado final de  $M_a$  para  $F_a^\cap$  e outra do estado final de  $M_a^R$  para  $F_a^\cap$ .

Deste modo,  $\delta_a^\cap$  é dada por:

	a	b	c	$\epsilon$
$\rightarrow$	$q_0^\cap$	$\emptyset$	$\emptyset$	$-, A$
$\star$	$F_a^\cap$	$\emptyset$	$\emptyset$	$\emptyset$
-A	AA,A-	BA,BB	CC	$F_a^\cap$
-B	AC	B-	CB	$\emptyset$
-C	AB,AA	BC	C-,CA	$\emptyset$
- -	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
AA	AA,A-	AA,AB	CC	$\emptyset$
AB	AC	A-	CB	$\emptyset$
AC	AB,AA	AC	C-,CA	$\emptyset$
A-	$\emptyset$	$\emptyset$	$\emptyset$	$F_a^\cap$
BA	CA,C-	AA,AB	BC	$\emptyset$
BB	CC	A-	BB	$\emptyset$
BC	CB,CA	AC	B-,BA	$\emptyset$
B-	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
CA	BA,B-	CA,CB	AC	$\emptyset$
CB	BC	C-	AB	$\emptyset$
CC	BB,BA	CC	A-,AA	$\emptyset$
C-	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

**Construindo os AFND's**  $M_b^\cap = M_b \cap M_b^R$  e  $M_c^\cap = M_c \cap M_c^R$

A estratégia usada para obter  $M_a^\cap$  a partir de  $M_a$  e  $M_a^R$  pode ser usada, *mutatis mutandis*, para obter  $M_b^\cap$  a partir de  $M_b$  e  $M_b^R$  e  $M_c^\cap$  a partir de  $M_c$  e  $M_c^R$ .

**Construindo o AFND**  $M = M_a^\cap \cup M_b^\cap \cup M_c^\cap$

Finalmente construímos o AFND  $M = (Q_M, \Sigma, \delta_M, q_{0M}, F_M)$  que reconhece  $L$  definindo:

$$M = M_a^\cap \cup M_b^\cap \cup M_c^\cap \quad (39)$$

Criamos um estado inicial  $q_{0M}$  para  $M$  a partir do qual há transições  $\epsilon$  para os estados iniciais de  $M_a^\cap$ ,  $M_b^\cap$  e  $M_c^\cap$ .

Também criamos um estado final  $FIM$  que recebe três transições  $\epsilon$  vindas dos estados finais de  $M_a^\cap$ ,  $M_b^\cap$  e  $M_c^\cap$ .

Assim, temos:

i)  $Q_M = \{q_{uv}^t | t \in \Sigma \wedge q_{uv} \in Q \times Q\} \cup \{q_{0M}\} \cup \{FIM\}$ .

ii)  $\delta_M$  é definida a por:

$$\delta_m(q_{0M}, \epsilon) = \{q_A^a, q_B^b, q_C^c\}$$

$$\delta_m(q_{uv}^t, x) = \{q_{wz}^t | q_{wz} \in \delta_t^\cap(q_{uv}, x) \forall q_{uv} \in Q_t^\cap \wedge \forall x \in \Sigma\}$$

iii)  $F_M = \{FIM, q_{0M}\}$ . O estado inicial  $q_{0M}$  é também um estado de aceitação pois a cadeia nula é um palíndromo.

c) Vamos criar um AFND  $M$  que aceita a linguagem  $L$ :

$$L = \{x \in \Sigma^* | x \text{ termina em '1aaaaaaaa' onde } a \in \Sigma\} \quad (40)$$

Podemos usar o não-determinismo para que  $M$  “adivinhe” que o caracter atualmente lido é o décimo da direita para a esquerda de  $x$ . Ou seja, para cada caracter lido de  $x$ ,  $M$  se desdobra em duas instância, cada uma percorrendo uma rota distinta:

i) Em uma destas rotas, o caracter lido é igual a 1 e é o décimo da direita para a esquerda. Portanto, o AFND transita do estado inicial  $\{-\}$  para o estado  $\{9\}$ . Se o AFND “adivinhou” corretamente, o AFND irá ler os próximos 9 caracteres 0 ou 1 transitando sucessivamente pelos estados  $\{8\}, \{7\}, \dots, \{1\}$  até chegar ao estado final  $\{0\}$  quando termina a leitura da cadeia  $x$  num estado de aceitação. Cada estado indica “quantos caracteres faltam até o fim da cadeia lida”.

- ii) Na outra rota, o caracter lido é um caracter normal. Neste caso, o AFND continua no estado inicial  $\{-\}$ .

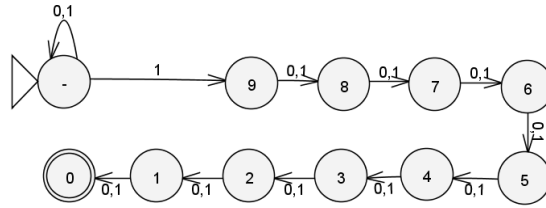


Figura 15 AFND  $M$  do Exercício 2.8(c).

Se a decisão inicial do AFND estiver errada então há duas hipóteses:

- Há *mais* de 10 caracteres até o fim da cadeia. Neste caso o automata trava e, portanto, não reconhece a cadeia.
- Há *menos* de 10 caracteres até o fim da cadeia. Neste caso o automata não atinge o estado final e, portanto, não reconhece a cadeia.

Porém, sempre que um caracter de  $x$  é lido, se este caracter for 1, haverá uma outra instância do AFND que irá transitar do estado inicial  $\{-\}$  para o estado  $\{9\}$ , tentando seguir a rota (i) acima descrita.

Portanto, num certo momento,  $M$  irá aceitar  $x$  sempre que  $x \in L$  (pois neste caso o AFND terá percorrido a rota (i) acima) e irá rejeitar  $x$  quando  $x \notin L$  (pois neste caso o AFND continuará na rota (ii) ou, tendo seguido a rota (i), irá parar em algum estado que não é de aceitação ou irá travar).

Note-se que este AFND tem menos estados e é muito mais fácil de obter e entender do que o AFD da questão 2.5(e).

**Exercício 2.9.** Construa os DFA's equivalentes aos NFA's.

- a)  $(\{p, q, r, s\}, \{0, 1\}, \delta_1, p, s)$  onde  $\delta_1$  é dada por:

		0	1
$\rightarrow$	$p$	$p, q$	$p$
	$q$	$r$	$r$
	$r$	$s$	—
$\star$	$s$	$s$	$s$

- b)  $(\{p, q, r, s\}, \{0, 1\}, \delta_2, p, \{q, s\})$  onde  $\delta_2$  é dada por:

		0	1
$\rightarrow$	$p$	$q, s$	$q$
$\star$	$q$	$r$	$q, r$
	$r$	$s$	$p$
$\star$	$s$	—	$p$

**Solução:**

- a) Se  $M = (Q = \{p, q, r, s\}, \Sigma = \{0, 1\}, \delta = \delta_1, q_0 = p, F = s)$  então o AFD correspondente é  $M' = (Q', \Sigma', \delta', q'_0, F')$ .

Sabemos que:

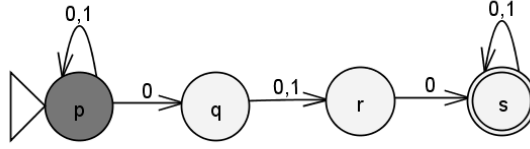


Figura 16 AFND  $M$  do Exercício 2.9(a).

- i)  $Q' = 2^Q = \{\emptyset, [p], [q], [r], [s], [pq], [pr], [ps], [qr], [qs], [rs], [pqr], [pqs], [prs], [qrs], [pqrs]\}$ .
- ii)  $\Sigma' = \Sigma$ .
- iii)  $q'_0 = [p]$
- iv)  $F' = \{[s], [ps], [qs], [rs], [pqs], [prs], [qrs], [pqrs]\}$

Para obter  $\delta'$  fazemos:

- 1)  $\delta(p, 0) = \{p, q\} \Rightarrow \delta'([p], 0) = \{[p, q]\}$
- 2)  $\delta(p, 1) = \{p\} \Rightarrow \delta'([p], 1) = \{[p]\}$
- 3)  $\delta(p, 0) = \{p, q\} \wedge \delta(q, 0) = \{r\} \Rightarrow \delta'([p, q], 0) = \{[p, q]\}$
- 4)  $\delta(p, 1) = \{p\} \wedge \delta(q, 1) = \{r\} \Rightarrow \delta'([p, q], 1) = \{[p, r]\}$
- 5)  $\delta(p, 0) = \{p, q\} \wedge \delta(r, 0) = \{s\} \Rightarrow \delta'([p, r], 0) = \{[p, q, s]\}$
- 6)  $\delta(p, 1) = \{p\} \wedge \delta(r, 1) = \emptyset \Rightarrow \delta'([p, r], 1) = \{[p]\}$
- 7)  $\delta(p, 0) = \{p, q\} \wedge \delta(q, 0) = \{r\} \wedge \delta(s, 0) = \{s\} \Rightarrow \delta'([p, q, s], 0) = \{[p, q, r, s]\}$
- 8)  $\delta(p, 1) = \{p\} \wedge \delta(q, 1) = \{r\} \wedge \delta(s, 1) = \{s\} \Rightarrow \delta'([p, q, s], 1) = \{[p, r, s]\}$
- 9)  $\delta(p, 0) = \{p, q\} \wedge \delta(q, 0) = \{r\} \wedge \delta(r, 0) = \{s\} \wedge \delta(s, 0) = \{s\} \Rightarrow \delta'([p, q, r, s], 0) = \{[p, q, r, s]\}$
- 10)  $\delta(p, 1) = \{p\} \wedge \delta(q, 1) = \{r\} \wedge \delta(r, 1) = \emptyset \wedge \delta(s, 1) = \{s\} \Rightarrow \delta'([p, q, r, s], 1) = \{[p, r, s]\}$
- 11)  $\delta(p, 0) = \{p, q\} \wedge \delta(r, 0) = \{s\} \wedge \delta(s, 0) = \{s\} \Rightarrow \delta'([p, r, s], 0) = \{[p, q, s]\}$
- 12)  $\delta(p, 1) = \{p\} \wedge \delta(r, 1) = \emptyset \wedge \delta(s, 1) = \{s\} \Rightarrow \delta'([p, r, s], 1) = \{[p, s]\}$
- 13)  $\delta(p, 0) = \{p, q\} \wedge \delta(s, 0) = \{s\} \Rightarrow \delta'([p, s], 0) = \{[p, q, s]\}$
- 14)  $\delta(p, 1) = \{p\} \wedge \delta(s, 1) = \{s\} \Rightarrow \delta'([p, s], 1) = \{[p, s]\}$

Portanto,  $\delta'$  é definida por:

		0	1
$\rightarrow$	$[p]$	$[p, q]$	$[p]$
	$[p, q]$	$[p, q]$	$[p, r]$
	$[p, r]$	$[p, q, s]$	$[p]$
*	$[p, q, s]$	$[p, q, r, s]$	$[p, r, s]$
*	$[p, q, r, s]$	$[p, q, r, s]$	$[p, r, s]$
*	$[p, r, s]$	$[p, q, s]$	$[p, s]$
*	$[p, s]$	$[p, q, s]$	$[p, s]$

Os demais estados de  $Q'$  não são acessíveis a partir de  $q'_0 = [p]$ .

- b) Se  $M = (Q = \{p, q, r, s\}, \Sigma = \{0, 1\}, \delta = \delta_1, q_0 = p, F = \{q, s\})$  então o AFD correspondente é  $M' = (Q', \Sigma', \delta', q'_0, F')$ .

Sabemos que:

- i)  $Q' = 2^Q = \{\emptyset, [p], [q], [r], [s], [pq], [pr], [ps], [qr], [qs], [rs], [pqr], [pqs], [prs], [qrs], [pqrs]\}$ .
- ii)  $\Sigma' = \Sigma$ .

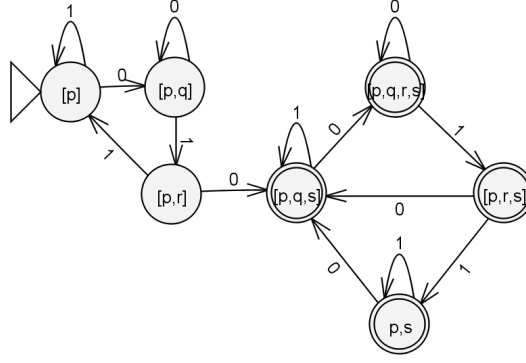


Figura 17 AFD  $M'$  do Exercício 2.9(a).

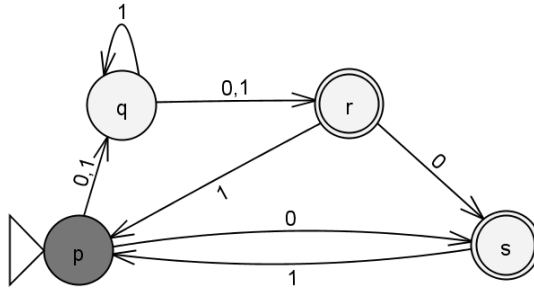


Figura 18 AFND  $M$  do Exercício 2.9(b).

iii)  $q'_0 = [p]$

iv)  $F' = \{[q], [s], [pq], [qr], [qs], [rs], [pqr], [pqs], [prs], [qrs], [pqrs]\}$

Para obter  $\delta'$  fazemos:

- 1)  $\delta(p, 0) = \{q, s\} \Rightarrow \delta'([p], 0) = \{[q, s]\}$
- 2)  $\delta(p, 1) = \{q\} \Rightarrow \delta'([p], 1) = \{[q]\}$
- 3)  $\delta(q, 0) = \{r\} \wedge \delta(s, 0) = \emptyset \Rightarrow \delta'([q, s], 0) = \{[r]\}$
- 4)  $\delta(q, 1) = \{q, r\} \wedge \delta(s, 1) = \{p\} \Rightarrow \delta'([q, s], 1) = \{[p, q, r]\}$
- 5)  $\delta(q, 0) = \{r\} \Rightarrow \delta'([q], 0) = \{[r]\}$
- 6)  $\delta(q, 1) = \{q, r\} \Rightarrow \delta'([q], 1) = \{[q, r]\}$
- 7)  $\delta(r, 0) = \{s\} \Rightarrow \delta'([r], 0) = \{[s]\}$
- 8)  $\delta(r, 1) = \{p\} \Rightarrow \delta'([r], 1) = \{[p]\}$
- 9)  $\delta(p, 0) = \{q, s\} \wedge \delta(q, 0) = \{r\} \wedge \delta(r, 0) = \{s\} \Rightarrow \delta'([p, q, r], 0) = \{[q, r, s]\}$
- 10)  $\delta(p, 1) = \{q\} \wedge \delta(q, 1) = \{q, r\} \wedge \delta(r, 1) = \{p\} \Rightarrow \delta'([p, q, r], 1) = \{[p, q, r]\}$
- 11)  $\delta(q, 0) = \{r\} \wedge \delta(r, 0) = \{s\} \Rightarrow \delta'([q, r], 0) = \{[r, s]\}$
- 12)  $\delta(q, 1) = \{q, r\} \wedge \delta(r, 1) = \{p\} \Rightarrow \delta'([q, r], 1) = \{[p, q, r]\}$
- 13)  $\delta(s, 0) = \emptyset \Rightarrow \delta'([s], 0) = \{\emptyset\}$
- 14)  $\delta(s, 1) = \{p\} \Rightarrow \delta'([s], 1) = \{[p]\}$
- 15)  $\delta(q, 0) = \{r\} \wedge \delta(r, 0) = \{s\} \wedge \delta(s, 0) = \emptyset \Rightarrow \delta'([q, r, s], 0) = \{[r, s]\}$

- 16)  $\delta(q, 1) = \{q, r\} \wedge \delta(r, 1) = \{p\} \wedge \delta(s, 1) = \{p\} \Rightarrow \delta'([q, r, s], 1) = \{[p, q, r]\}$   
 17)  $\delta(r, 0) = \{s\} \wedge \delta(s, 0) = \emptyset \Rightarrow \delta'([r, s], 0) = \{[s]\}$   
 18)  $\delta(r, 1) = \{p\} \wedge \delta(s, 1) = \{p\} \Rightarrow \delta'([r, s], 1) = \{[p]\}$

Portanto,  $\delta'$  é definida por:

		0	1
$\rightarrow$	$[p]$	$[q, s]$	$[q]$
$\star$	$[q]$	$[r]$	$[q, r]$
	$[r]$	$[s]$	$[p]$
$\star$	$[s]$	$[\emptyset]$	$[p]$
$\star$	$[q, r]$	$[r, s]$	$[p, q, r]$
$\star$	$[q, s]$	$[r]$	$[p, q, r]$
$\star$	$[r, s]$	$[s]$	$[p]$
$\star$	$[p, q, r]$	$[q, r, s]$	$[p, q, r]$
$\star$	$[q, r, s]$	$[r, s]$	$[p, q, r]$
	$[\emptyset]$	$[\emptyset]$	$[\emptyset]$

Os demais estados de  $Q'$  não são acessíveis a partir de  $q'_0 = [p]$ .

**Exercício 2.10.** Construa expressões regulares para cada uma das seguintes linguagens sobre o alfabeto  $\Sigma = \{0, 1\}$ . Justifique porque sua expressão regular é correta.

- a) O conjunto de todas as cadeias com não mais do que um par de 0's consecutivos e não mais do que um par de 1's consecutivos.  
 b) O conjunto de todas as cadeias em que todo par de 0's consecutivos aparece antes de qualquer par de 1's consecutivos.  
 c) O conjunto de todas as cadeias que não contenham 101 como subcadeia.  
 d) O conjunto de todas as cadeias com igual número de 0's e 1's tal que nenhum prefixo tem dois ou mais 0's que 1's e nem dois ou mais 1's que 0's.

**Solução:**

- a) Vamos começar com a cadeia nula, que pertence à linguagem, e ir completando a expressão regular. Comentários justificam cada passo.
- i) Cadeia nula pertence à linguagem:  $\epsilon$
  - ii) Podemos ter a cadeia nula ou um 0:  $(\epsilon + 0)$
  - iii) A cadeia acima pode ser seguida por um número qualquer de 01's. Deste modo, garantimos que não aparecerão 00's nem 11's:  $(\epsilon + 0)(10)^*$
  - iv) Agora vamos permitir que apareça no máximo um 00:  $(\epsilon + 0)(10)^*(\epsilon + 0)$
  - v) Em seguida, vamos deixar que ocorra um número qualquer de 01's, garantindo que não haverá novos 00's nem 11's:  $(\epsilon + 0)(10)^*(\epsilon + 0)(10)^*$ .
  - vi) Agora vamos permitir que ocorra um 11 mas sem que este seja seguido por outro 00 ou outro 11:  $(\epsilon + 0)(10)^*(\epsilon + 0)(10)^*(\epsilon + 1)(10)^*$
  - vii) Finalmente, permitimos que a cadeia termine em 0 ou 1:  $(\epsilon + 0)(10)^*(\epsilon + 0)(10)^*(\epsilon + 1)(10)^*(\epsilon + 1)$

A última expressão regular, acima, define cadeias nas quais o 00 ocorre antes do 11. Usando um raciocínio simétrico, definimos esta expressão regular na qual o 11 ocorre antes do 00:  $(\epsilon + \mathbf{0})(\mathbf{10})^*(\epsilon + \mathbf{1})(\mathbf{10})^*(\epsilon + \mathbf{0})(\mathbf{10})^*(\epsilon + \mathbf{0})$

Queremos ter até um 00 e até um 11 em qualquer posição relativa um ao outro. Portanto, a expressão regular procurada é a soma das duas últimas expressões regulares mostradas acima:

$$(\epsilon + \mathbf{0})(\mathbf{10})^*(\epsilon + \mathbf{0})(\mathbf{10})^*(\epsilon + \mathbf{1})(\mathbf{10})^*(\epsilon + \mathbf{1}) + (\epsilon + \mathbf{0})(\mathbf{10})^*(\epsilon + \mathbf{1})(\mathbf{10})^*(\epsilon + \mathbf{0})(\mathbf{10})^*(\epsilon + \mathbf{0})$$

- b) A expressão  $(\mathbf{0} + \mathbf{10})^*$  gera cadeias em que 11 não aparece e 00 pode eventualmente aparecer.

Analogamente,  $(\mathbf{1} + \mathbf{01})^*$  gera cadeias em que 00 não aparece e 11 pode eventualmente aparecer.

Para garantir que 11 apareça sempre após 00, basta concatenar as duas expressões acima:

$$(\mathbf{0} + \mathbf{10})^*(\mathbf{1} + \mathbf{01})^*$$

- c) Vamos começar com a cadeia nula, que pertence à linguagem, e ir completando a expressão regular até obter a expressão mais genérica. Comentários justificam cada passo.

- i) Cadeia nula pertence à linguagem:  $\epsilon$
- ii) Podemos ter a cadeia nula ou um 0:  $(\epsilon + \mathbf{0})$
- iii) Na verdade, podemos ter a cadeia nula ou um número qualquer de 0's:  $\mathbf{0}^*$
- iv) A cadeia acima pode ser seguida por um número qualquer de 1's:  $\mathbf{0}^*\mathbf{1}^*$
- v) Estes 1's podem ser seguidos por um número par de 0's:  $\mathbf{0}^*(\mathbf{1} + \mathbf{00})^*$
- vi) Finalmente, a cadeia pode ser terminada por um número qualquer de 0's:

$$\mathbf{0}^*(\mathbf{1} + \mathbf{00})^*\mathbf{0}^*$$

- d) Vamos começar com a cadeia nula, que pertence à linguagem, e ir completando a expressão regular até obter a expressão mais genérica. Comentários justificam cada passo.

- i) Cadeia nula pertence à linguagem:  $\epsilon$
- ii) Para manter o mesmo número de 0's e 1's podemos ter a cadeia nula ou várias instâncias de 01's:  $(\mathbf{01})^*$
- iii) Na cadeia acima, cada prefixo tem no máximo um 0 a mais que o número de 1's. Falta incluir os prefixos que têm no máximo um 1 a mais que o número de 0's:

$$(\mathbf{01} + \mathbf{10})^*$$

**Exercício 2.11.** *Descreva, em português, os conjuntos definidos pelas seguintes expressões regulares.*

- a)  $(\mathbf{11} + \mathbf{0})^*(\mathbf{00} + \mathbf{1})^*$
- b)  $(\mathbf{1} + \mathbf{01} + \mathbf{001})^*(\epsilon + \mathbf{0} + \mathbf{00})$
- c)  $[\mathbf{00} + \mathbf{11} + (\mathbf{01} + \mathbf{10})(\mathbf{00} + \mathbf{11})^*(\mathbf{01} + \mathbf{10})]^*$

**Solução:**

- a) Conjunto de cadeias binárias que contêm um número par de 1's seguido por um número par de 0's. Consideramos que três zeros seguidos podem ser interpretados como duas cadeias 00 seguidas.

- b) Conjunto de cadeias binárias que contêm não mais do que 2 zeros consecutivos.
- c) Conjunto de cadeias binárias de comprimento par, em cujos prefixos o número de zeros não difere mais do que uma unidade do número de uns.

**Exercício 2.12.** Construa o autômata finito equivalente às seguintes expressões regulares.

a)  $10 + (0 + 11)0^*1$

b)  $01[(10)^* + 111)^* + 0]^*1$

c)  $((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*$

**Solução:**

A solução está nas figuras 19 a 21.

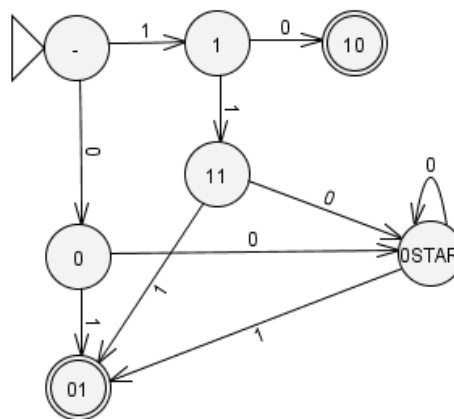


Figura 19 AFND do Exercício 2.12(a).

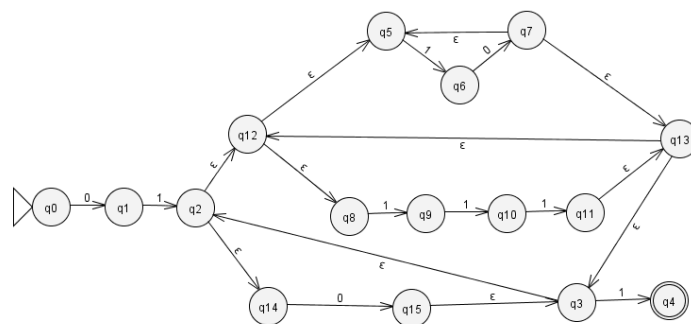


Figura 20 AFND do Exercício 2.12(b).



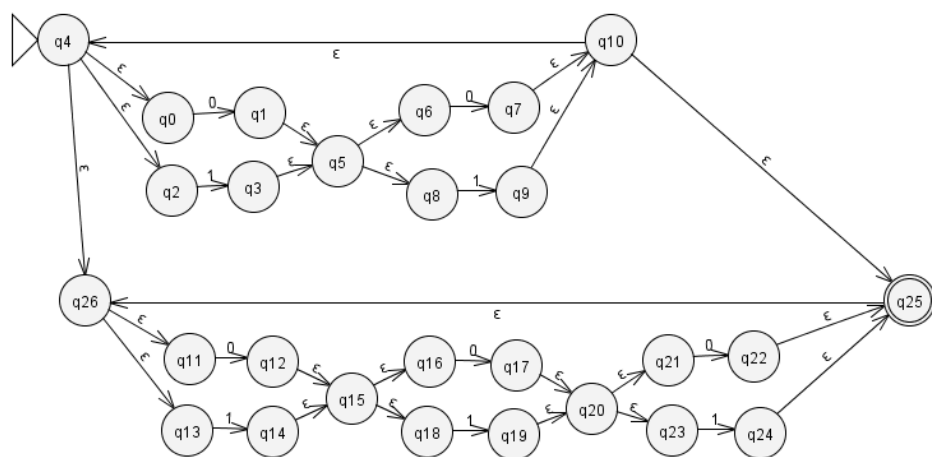


Figura 21 AFND do Exercício 2.12(c).

**Exercício 2.13.** Construa as expressões regulares para os diagramas de estado da Figura 22.

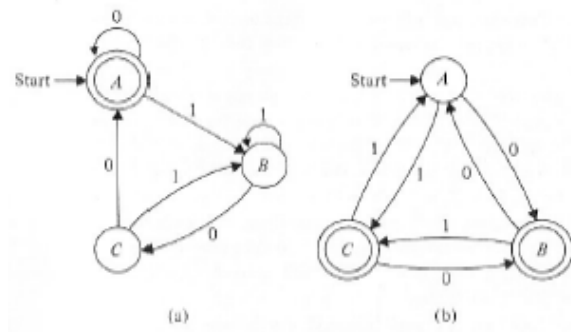


Figura 22 Diagrama de transição para o Exercício 2.13.

### Solução:

Inicialmente, criamos uma cópia do automato para cada estado final.

Em seguida, criamos dois estados  $q_i$  e  $q_f$  que substituem os estados inicial do AFD. De  $q_i$  sai uma transição  $\epsilon$  para o antigo estado inicial e em  $q_f$  chegam transições  $\epsilon$  do(s) antigo(s) estado(s) final(is).

Em seguida, vamos removendo os estados e substituindo as transições por expressões regulares equivalentes até que a expressão regular final surge no arco entre  $q_i$  e  $q_f$ .

Alguns passos estão detalhados nas figuras 23 a 27.

a) A expressão regular da Figura 22 (a) é:

$$0^* + (11^*(01)^*00)^*$$

b) A expressão regular da Figura 22 (b) é:

$$(11)^*0(10)^* + (00)^*1(01)^*$$

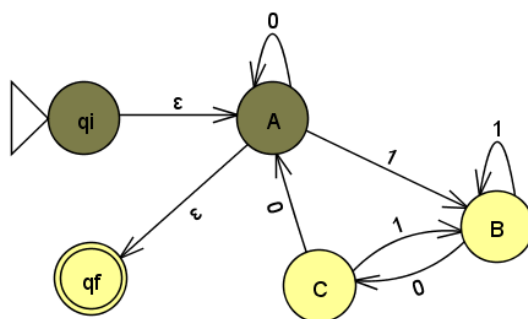


Figura 23 Passo 1 do ex. 2.12(a): Inclusão dos estados  $q_i$  e  $q_f$ .

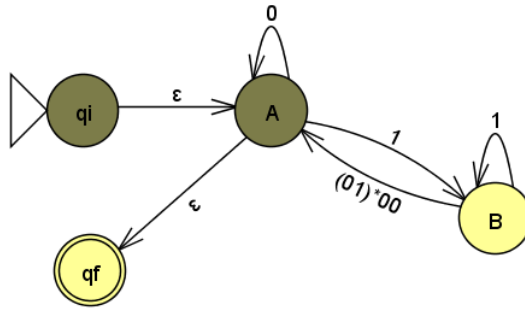


Figura 24 Passo 2 do ex. 2.12(a): Remoção do Estado  $C$ .

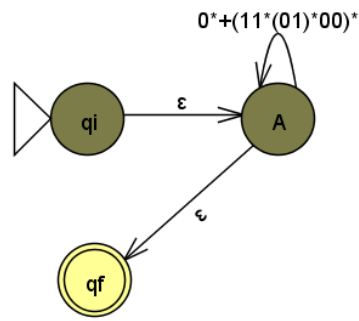


Figura 25 Passo 3 do ex. 2.12(a): Remoção do Estado  $B$ .

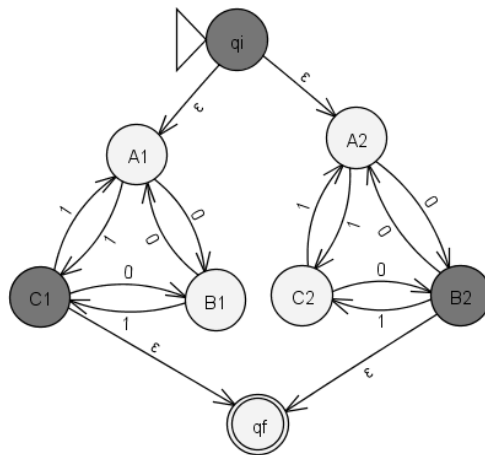


Figura 26 Passo 1 do ex. 2.12(b): Uma cópia para cada estado final mais inclusão dos estados  $q_i$  e  $q_f$ .

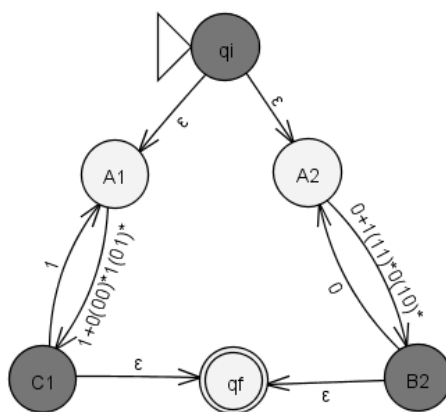


Figura 27 Passo 2 do ex. 2.12(b): Remoção dos estados  $B1$  e  $C2$ .

---

**Exercício 2.14.** Use as idéias da prova do teorema 2.4 do Hopcroft<sup>1</sup> para construir algoritmos para os seguintes problemas:

- a) Encontre o caminho de custo mínimo entre dois vértices em um grafo direcionado onde cada vértice é associado a um custo não negativo.
- b) Determine o número de cadeias de comprimento  $n$  aceito por um automata finito.

**Solução:**

Nas condições do teorema 2.4, temos um autômata finito  $M$  de  $n$  estados definido como  $M = (\{q_1, q_2, \dots, q_n\}, \Sigma, \delta, q_1, F)$

O teorema mostra que, para todo  $\{i, j, k\} \subset \mathbb{N}$  com  $i \leq n$ ,  $j \leq n$  e  $k \leq n$ , temos:

- i)  $R_{ij}^k$  é o conjunto de todas as cadeias  $x$  que levam  $M$  do estado  $q_i$  para o estado  $q_j$  tal que, se numerarmos os estados  $q_1, q_2, \dots, q_n$  como  $1, 2, \dots, n$ , respectivamente, nenhum dos estados intermediários tem número maior que  $k$ .
- ii) O valor de  $R_{ij}^k$  é definido recursivamente por:

$$R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1} \quad (41)$$

$$R_{ij}^0 = \begin{cases} \{a | \delta(q_i, a) = q_j\} & \text{se } i \neq j \\ \{a | \delta(q_i, a) = q_j\} \cup \{\epsilon\} & \text{se } i = j \end{cases} \quad (42)$$

- a) Mantemos as definições de  $M, i, j$  e  $k$  tal como definidos acima.

Definimos  $C_{ij}$  como o custo associado ao arco que liga os estados  $q_i$  e  $q_j$ .

Como os custos são não negativos temos que  $C_{ij} \geq 0$  para todo  $i, j$ .

Mais formalmente:

$$C(i, j) = \begin{cases} \text{custo do arco } i \rightarrow j & \text{se } i \neq j \wedge \delta(q_i, a) = q_j \neq \emptyset \\ \infty & \text{se } i \neq j \wedge \delta(q_i, a) = \emptyset \\ 0 & \text{se } i = j \end{cases} \quad (43)$$

Para o cálculo do custo mínimo, redefinimos  $R_{ij}^k$  com  $k \leq n$ ,  $i \leq n$  e  $j \leq n$ , recursivamente como:

$$R_{ij}^k = \min(R_{ij}^{k-1}, R_{ik}^{k-1} + R_{kj}^{k-1}) \quad (44)$$

$$R_{ij}^0 = C(i, j) \quad (45)$$

**Demonstração.** Vamos provar, por indução em  $k$ , que a definição acima define o custo mínimo para todo  $k \leq n$ .

*Base:* Considerando  $k = 0$ .

Pela definição do Teorema 2.4 (ver item (i) acima) e pela eq. 45,  $R_{ij}^0$  mede o custo entre os nós  $q_i$  e  $q_j$  tal que os caminhos intermediários tenham numeração igual a 0. Como os estados estão numerados de 1 a  $n$ , temos que não há estados com numeração menor que 1. Portanto,  $R_{ij}^0$  mede o custo entre os nós  $q_i$  e  $q_j$  considerando a ligação “direta” (sem estados intermediários) entre eles.

Temos que considerar 3 casos:

---

<sup>1</sup>HOPCROFT, John E. & ULLMAN, Jeffrey D., Introduction to Automata Theory, Languages and Computation., 2nd. ed., 1979, ISBN 0-201-02988-X Pág. 33

- 1)  $i = j$ : Neste caso o custo de ir para o mesmo estado em que se está é zero. Pela eq. 43,  $i = j \Rightarrow C(i, j) = 0$ . Portanto, pela eq. 45,  $R_{ij}^0 = C(i, j) = 0$ .
- 2)  $i \neq j$  e há uma conexão entre  $q_i$  e  $q_j$ : Neste caso o custo de ir de  $q_i$  para  $q_j$  é  $C(i, j)$ . E, de fato, pela eq. 45,  $R_{ij}^0 = C(i, j)$ .
- 3)  $i \neq j$  e não há uma conexão entre  $q_i$  e  $q_j$ : Neste caso o custo de ir de  $q_i$  para  $q_j$  é indefinido. Como estamos buscando o custo mínimo, podemos considerá-lo infinito. E, de fato, pela eq. 43 temos que  $C(i, j) = \infty$  e, portanto, pela eq. 45,  $R_{ij}^0 = \infty$ .

Portanto, as equações de custo valem para  $k = 0$ .

*Hipótese Indutiva:* Considerando que as equações valem para  $k - 1$  vamos mostrar que valem para  $k$ .

De fato, se as equações valem para  $k - 1$  então  $R_{ij}^{k-1}$  contém o comprimento mínimo entre  $q_i$  e  $q_j$  considerando como estados intermediários aqueles de numeração menor ou igual a  $k - 1$ .

Pela mesma hipótese, estão definidos os custos mínimos, considerando estados intermediários de numeração menor ou igual a  $k - 1$ :

- i)  $R_{ik}^{k-1}$  que é o custo de  $q_i$  até  $q_k$ .
- ii)  $R_{kj}^{k-1}$  que é o custo de  $q_k$  até  $q_j$ .

Para obter  $R_{ij}^k$ , que é o custo de  $q_i$  para  $q_j$  passando por estados intermediários de numeração menor ou igual a  $k$ , temos dois caminhos:

- 1) Ir de  $q_i$  para  $q_j$  passando pelos estados de numeração menor ou igual a  $k - 1$  o que resulta em  $R_{ij}^{k-1}$
- 2) Ir de  $q_i$  até  $q_k$  gastando  $R_{ik}^{k-1}$  e depois ir de  $q_k$  até  $q_j$  gastando  $R_{kj}^{k-1}$ . Neste caso, a distância é  $R_{ik}^{k-1} + R_{kj}^{k-1}$ .

Na definição de  $R_{ij}^k$ , a função **min** considera o menor valor dentre os listados nos itens (1) e (2) acima. Portanto, garantimos obter o caminho de menor custo.

Portanto, pelo Princípio da Indução Finita, as equações 43 a 45 valem para todo  $k \in \mathbb{N}$  tal que  $k \leq n$ .  $\square$

b) A resolução é semelhante à do item (a) acima.

Seja  $(R_{ij}^k)_m$  o número de caminhos do estado  $i$  para o estado  $j$ , de comprimento  $m$ , tal que, nos caminhos intermediários, não haja nenhum estado com numeração maior que  $k$ .

Por indução em  $k$ , mostramos que é possível calcular  $m$  para quaisquer  $i, j$  e para  $m \leq n$ .

O número de cadeias de comprimento  $n$  aceito pelo automata finito é dado por  $n = \sum_{j=1}^{n_a} \sum_{i=1}^j (R_{ij}^k)_m$  onde 1 é o estado inicial,  $j$  é um estado de aceitação e  $n_a$  é o número de estados de aceitação.

**Exercício 2.15.** Construa um AFND equivalente a um 2AFD  $(\{q_0, \dots, q_4\}, \{0, 1\}, \delta, q_0, \{q_2\})$  em que  $\delta$  é dado por:

	0	1
$q_0$	$(q_0, R)$	$(q_1, R)$
$q_1$	$(q_1, R)$	$(q_2, R)$
$q_2$	$(q_2, R)$	$(q_3, L)$
$q_3$	$(q_4, L)$	$(q_3, L)$
$q_4$	$(q_0, R)$	$(q_4, L)$

**Solução:**

Inicialmente, desenhamos o 2AFD obtendo a Figura 28. Em seguida, obtivemos o AFND equivalente de acordo com as considerações abaixo.

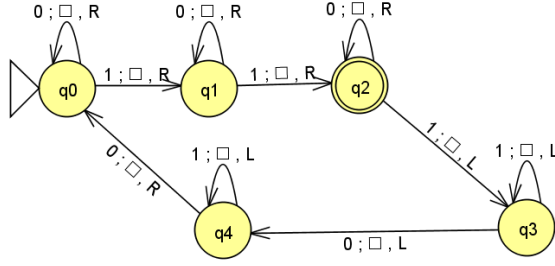


Figura 28 2AFD do Exercício 2.15.

Seja  $M$  o 2AFD da Figura 28.

Seja  $w \in L(M)$  uma certa cadeia reconhecida por  $M$ .

Como o 2AFD é bi-direcional, pode acontecer que, ao aceitar  $w$  este automato passe várias vezes sobre um certo trecho  $t$  de  $w$ . Isto é,  $M$  está num certo estado  $q_i$  quando começa a ler  $t$  e vai para um estado  $q_j = \hat{\delta}(q_i, t)$  após ter lido  $t$ . Mais adiante, pode ser que  $M$  retorne ao início de  $t$ , porém agora num estado  $q_k$  e vá para o estado  $q_m = \hat{\delta}(q_k, t)$ .

Ou seja,  $M$  pode ler  $t$  várias vezes. Em cada vez,  $M$  inicia a leitura num certo estado ( $q_i$  e  $q_k$  em nosso exemplo) e vai para um outro estado ( $q_j$  e  $q_m$  em nosso exemplo).

Seja  $M'$  um AFND equivalente a  $M$ .

Como  $M'$  é unidirecional, a leitura do trecho  $t$  tem que ser feita uma única vez. Portanto, para manter a equivalência,  $M'$  tem que chegar no início de  $t$  estando simultaneamente nos estados  $q_i$  e  $q_k$ . Deste modo, uma instância de  $M'$  irá chegar em  $q_j$  e a outra instância chegará em  $q_m$ .

Impossibilitado de ler  $t$  várias vezes,  $M'$  utiliza transições  $\epsilon$  para, ao começar ler  $t$ , estar em todos os estados que alcançaria lendo  $t$  várias vezes.

Analisando a Figura 28, notamos que as transições entre os estados  $q_0$ ,  $q_1$  e  $q_2$  são feitas lendo a cadeia de entrada da esquerda para a direita. Portanto, o AFND equivalente terá exatamente as mesmas transições. Para facilitar, repetimos até mesmo o nome dos estados.

Quando  $M$  está em  $q_2$  e lê 1 na cadeia de entrada, ele transita para  $q_3$  e começa a ler a cadeia da direita para a esquerda. A única forma de, saindo do estado  $q_2$ , transitar por  $q_3$  e  $q_4$  voltando novamente ao estado  $q_0$  é, lendo da direita para esquerda, encontrar caracteres na forma  $x = 11^*01^*0$ .

Para reproduzir este comportamento,  $M'$ , que só pode ler da esquerda para a direita, tem que seguir duas instâncias diferentes a partir de  $q_0$ :

- i)  $M'$  vai pelo caminho  $q_0, q_1$  e  $q_2$  lendo cadeias na forma  $y = 0^*10^*10^*$
- ii)  $M'$  vai pelo caminho  $q_4, q_3$  e  $q_2$  lendo cadeias na forma  $x^R = 01^*01^*1$

Para simular este comportamento, criamos um novo estado inicial  $s_0$  com transições  $\epsilon$  para as alternativas acima, originando o NFA da Figura 29.

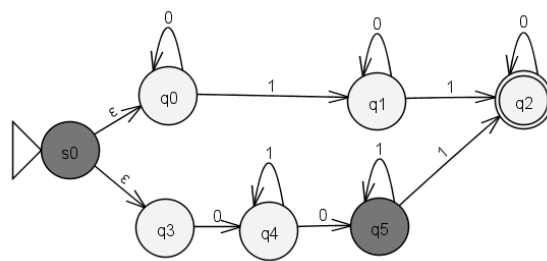


Figura 29 NFA do Exercício 2.15.



---

**Exercício 2.24.** Encontre as máquinas de Mealy e de Moore para os seguintes processos:

- a) Para entradas de  $(0 + 1)^*$ , se a entrada termina em 101, imprima A; se termina em 110, imprima B e nos demais casos imprima C.
- b) Para entradas de  $(0 + 1 + 2)^*$ , imprima o resíduo módulo 5 da entrada tratada como um número ternário (base 3 com dígitos 0, 1 e 2).

**Solução:**

- a) Ver Figuras 30 e 31.

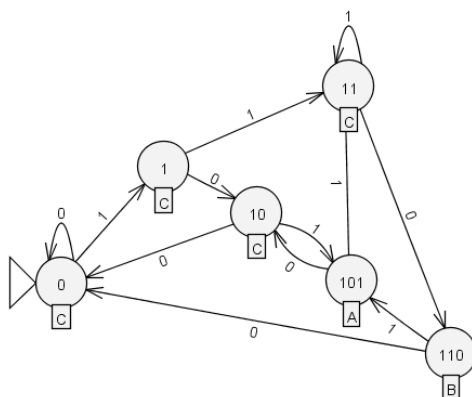


Figura 30 Máquina de Moore do Exercício 2.24(a).

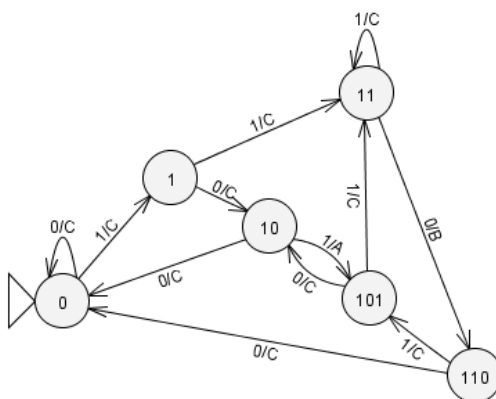


Figura 31 Máquina de Mealy do Exercício 2.24(a).

- b) Seja  $x$  uma cadeia de entrada das máquinas. Seja  $n = |x|$ .

Então  $x$  é da forma  $x = d_1 d_2 \dots d_{n-1} d_n$  onde  $d_i \in \Sigma = \{0, 1, 2\}$  e o valor decimal de  $x$  é dado pela expressão  $V(x) = \sum_{k=1}^n d_k 3^{n-k}$ .

Por exemplo, se  $x = 211$  então  $V(x) = 2 \times 3^2 + 1 \times 3^1 + 1 \times 3^0$ .

O resto da divisão de  $V(x)$  por 5 é um número natural  $r$  tal que  $0 \leq r \leq 4$ .

No início, a máquina não leu nenhum carácter de  $x$  e temos  $r_0 = 0$ . À medida que vamos lendo o  $i$ -ésimo dígito de  $x$  (que chamamos de  $d_i$ ), temos que, para  $1 \leq i \leq n$ :

$$r_i = (r_{i-1} \times 3 + d_i) \bmod 5 \quad (46)$$

Usando este resultado, construímos as máquinas Moore e Mealy das Figuras 30 e 31 onde o nome do estado é o valor do resíduo.

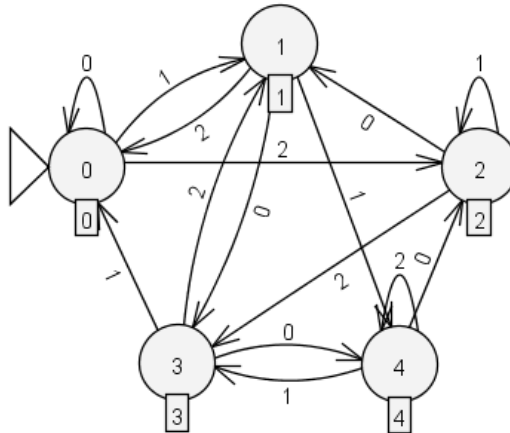


Figura 32 Máquina de Moore do Exercício 2.24(b).

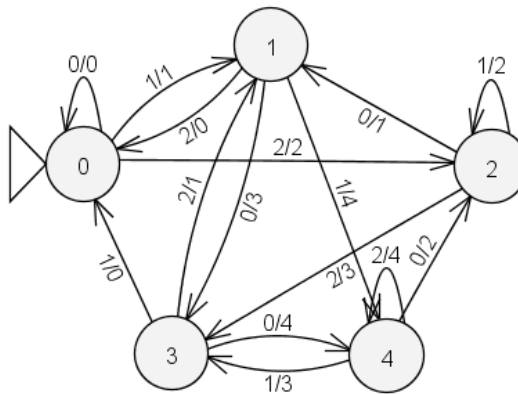


Figura 33 Máquina de Mealy do Exercício 2.24(b).

---

**Exercício 3.1.** *Quais dentre as seguintes linguagens são regulares? Prove sua resposta.*

- a)  $\{0^{2n} | n \geq 1\}$
- b)  $\{0^m 1^n 0^{m+n} | m \geq 1 \wedge n \geq 1\}$
- c)  $\{0^n | n \text{ é primo}\}$
- d) *O conjunto de todas as cadeias que não têm três 0's consecutivos.*

- e) O conjunto de todas as cadeias com igual número de 0's e 1's.
- f)  $\{x|x \in (\mathbf{0} + \mathbf{1})^* \wedge x = x^R\}$  onde  $x^R$  é a cadeia  $x$  escrita de trás para frente. Exemplo:  $(011)^R = 110$ .
- g)  $\{xwx^R|x, w \in (\mathbf{0} + \mathbf{1})^+\}$ .
- h)  $\{xx^Rw|x, w \in (\mathbf{0} + \mathbf{1})^+\}$ .

### Solução:

- a) A linguagem  $L = \{0^{2n}|n \geq 1\}$  aceita cadeias com número par de zeros, mas não aceita a cadeia nula. Portanto,  $L$  equivale à expressão regular  $(\mathbf{00})(\mathbf{00})^*$ . Logo,  $L$  é regular.
- b) Se  $L = \{0^m1^n0^{m+n}|m \geq 1 \wedge n \geq 1\}$  então  $L$  é não regular

**Demonstração.** Pelo teorema de Myhill-Nerode.

Considere as classes  $[0^i]$  e  $[0^j]$  com  $\{i, j\} \subset \mathbb{N}$  e  $i \neq j$ .

Se  $i = m + n$  então  $L$  distingue  $[0^i]$  de  $[0^j]$  pois  $0^m1^n0^i \in L$  e  $0^m1^n0^j \notin L$ .

Como  $i$  e  $j$  são genéricos, existem infinitas classes de equivalência distintas entre si. Portanto, pelo teorema de Myhill-Nerode,  $L$  não é regular.  $\square$

- c) A linguagem  $L = \{0^n|n \text{ é primo}\}$  não é regular.

**Demonstração.** Por contradição.

Se, por absurdo,  $L$  é regular, então  $L$  é aceita por um AFD  $M$ . Seja  $s$  o número de estados de  $M$ . Considere um número primo  $n > s$ . Note que  $x = 0^n \in L$  e que  $|x| = n > s$ . Portanto, nas condições do Lema do Bombeamento, podemos escrever  $x = uvw$  com  $|v| > 0$  e  $|vw| < s$  tal que  $y = uv^k w \in L$  para todo  $k \in \mathbb{N}$ .

Seja  $i = |u| + |w|$  e  $j = |v|$ . Se  $y = uv^k w \in L$  então  $y = 0^{i+kj} \in L$ . Logo, pela definição de  $L$ ,  $i + kj$  é primo.

Se  $k = 0$  então  $i + kj = i + 0 \times j = i$  é primo.

Se  $k = i$  então  $i + kj = i + i \times j = i(1 + j)$  é primo.

Mas  $i(1 + j)$  não pode ser primo pois, como  $j = |v| \geq 1$  então  $(1 + j) > 1$  e, portanto,  $i(1 + j)$  é múltiplo de  $i$ , que é primo. Um número não pode ser, ao mesmo tempo, primo e múltiplo de um número primo.

Devido a esta contradição,  $L$  não é regular.  $\square$

- d) Se  $L$  é o conjunto de todas as cadeias que não têm três 0's consecutivos, então  $L$  é equivalente à expressão regular  $(\mathbf{1} + \mathbf{01} + \mathbf{001})^*(\epsilon + \mathbf{0} + \mathbf{00})$ . Portanto,  $L$  é regular.
- e) Se  $L$  é o conjunto de todas as cadeias com igual número de 0's e 1's então  $L$  não é regular.

**Demonstração.** Por contradição.

Se, por absurdo,  $L$  é regular, então  $L$  é aceita por um AFD  $M$ . Seja  $s$  o número de estados de  $M$ .

Se  $L$  reconhece todas as cadeias binárias com igual número de 0's e 1's então, em particular,  $L$  reconhece cadeias na forma  $0^n1^n$  para todo  $n \in \mathbb{N}$ .

Logo, se  $x = 0^p1^p$  com  $p = s$ , então  $x \in L$ .

Como  $|x| = 2p > s$ , podemos aplicar o Lema do Bombeamento fazendo  $x = uvw$  com  $|v| > 0$  e  $|uv| < s$  tal que  $y = uv^k w \in L$  para todo  $k \in \mathbb{N}$ .

Mas, se  $|uv| < s$  e  $p = s$ , então  $|uv| < p$ .

Se  $x = 0^p1^p$ ,  $x = uvw$  e  $|uv| < p$  então  $uv$  é formada apenas por caracteres 0. Logo, as cadeias  $u$  e  $v$  são formadas apenas por caracteres 0.

Mas se  $v$  é formada apenas por caracteres 0 e  $y = uv^k w$  então  $y$  possui mais caracteres 0's do que 1's para qualquer  $k > 1$ . Logo  $y \notin L$ , o que é uma contradição com o Lema do Bombeamento.

Devido a esta contradição,  $L$  não é regular. □

- f) Se  $L = \{x|x \in (\mathbf{0} + \mathbf{1})^* \wedge x = x^R\}$  onde  $x^R$  é a cadeia  $x$  escrita de trás para frente, então  $L$  é não regular.

**Demonstração.** Por contradição.

Se, por absurdo,  $L$  é regular. Então  $L$  é aceita por um AFD  $M$ . Seja  $s$  o número de estados de  $M$ .

Considere a cadeia  $z = 0^s 10^s$ . Por definição,  $z \in L$ .

Como,  $|z| \geq s$  podemos usar o Lema do Bombeamento, fazendo  $z = uvw$  com  $|v| > 0$  e  $|uv| < s$  tal que  $y = uv^k w \in L$  para todo  $k \in \mathbb{N}$ .

Sem perda de generalidade, fazemos:

- i)  $u = 0^r$  para  $0 \leq r < s$
- ii)  $v = 0^k$  tal que  $r + k = s$

Portanto, temos  $w = 10^s$ .

Se  $k = 2$  então  $y = uv^k w = uv^2 w = 0^r 0^{2k} 10^s = 0^{r+2k} 10^s$ .

Porém, como  $r + k = s$  temos que  $r + 2k > s$  e, portanto:

- i) Se  $|y|$  é par, então o lado esquerdo de  $y$  tem mais zeros que o lado direito.
- ii) Se  $|y|$  é ímpar, então o lado direito de  $y$  contém o dígito 1 e o lado esquerdo não contém.

Logo,  $y \notin L$  o que é uma contradição com o Lema do Bombeamento.

Devido a esta contradição,  $L$  não é regular. □

- g) A linguagem  $L = \{xwx^R|x, w \in (\mathbf{0} + \mathbf{1})^+\}$  é regular. Como  $x$  e  $w$  podem ser cadeias quaisquer, esta linguagem representa as cadeias binárias que começam e terminam com o mesmo caracter, a qual é equivalente à expressão regular  $\mathbf{0}(\mathbf{0} + \mathbf{1})^* \mathbf{0} + \mathbf{1}(\mathbf{0} + \mathbf{1})^* \mathbf{1}$ .

- h) Se  $L = \{xx^R w|x, w \in (\mathbf{0} + \mathbf{1})^+\}$ , então  $L$  não é regular.

**Demonstração.** Por contradição.

Vamos usar uma extensão do *Lema do Bombeamento* provada no exercício 3.2 logo abaixo.

Se, por absurdo,  $L$  é regular então  $L$  é aceita por um AFD  $M$ . Seja  $n$  o número de estados de  $M$ .

Seja  $\alpha = xx^R w$  com  $x = 010^n 1$  e  $w = \epsilon$ . Portanto,  $\alpha \in L$ .

Fazemos  $\alpha = 010^n 110^n 10 = z_1 z_2 z_3$  com  $z_1 = 01$ ,  $z_2 = 0^n$  e, portanto,  $z_3 = 110^n 10$ .

Fazemos  $z_2 = 0^n = uvw$  com  $|v| > 0$  e  $|uv| < n$ , atendendo às condições da extensão do *Lema do Bombeamento*.

Portanto, pela extensão do *Lema do Bombeamento*, para todo  $k \geq 0$  e  $y = uv^k w$  devemos ter  $\beta = z_1 y z_3 \in L$ .

Se  $k = 2$  então  $y = uv^2 w$ . Portanto,  $\beta = z_1 y z_3 = 010^{n+|v|} 110^n 10$ .

Mas se  $\beta \in L$  então  $\beta = pp^R q$  com  $\{p, q\} \subset (\mathbf{0} + \mathbf{1})^*$ .

Porém, existem apenas duas ocorrências da cadeia 010 em  $\beta$ : uma vez no prefixo e outra no sufixo. Considerando  $q = \epsilon$  temos que o prefixo de  $p$  tem que ser 010 e o sufixo de  $p^R$  tem que ser 010.

Em outras palavras, temos  $\beta = z_1 y z_3 = 010^{n+|v|} 110^n 10 = pp^R$ . Porém isso é impossível pois  $|v| > 0$  e, portanto,  $n + |v| \neq n$ .

Logo:

- i) Se  $|\beta|$  é par, o lado esquerdo de  $\beta$  fica com  $|v|$  zeros a mais que o lado direito.
- ii) Se  $|\beta|$  é ímpar, o lado esquerdo de  $\beta$  fica com menos um dígito 1 que o lado direito.

Portanto,  $\beta \notin L$  o que é uma contradição com a extensão do *Lema do Bombeamento*.

Devido a esta contradição,  $L$  não é regular. □

**Exercício 3.2.** Prove a seguinte extensão do Lema do Bombeamento para linguagens regulares. Seja  $L$  uma linguagem regular. Então existe uma constante  $n$  tal que, para cada  $z_1, z_2, z_3$ , com  $z_1 z_2 z_3 \in L$  e  $|z| = n$ ,  $z_2$  pode ser escrito como  $z_2 = uvw$  tal que  $|v| \geq 1$  e, para cada  $i \geq 0$ ,  $z_1 u v^i w z_3 \in L$ .

**Solução:**

**Demonstração.** Por construção.

Se  $L$  é regular então  $L$  é aceita por um AFD  $M = (Q, \Sigma, \delta, q_0, F)$ .

Seja  $n = |Q|$  e  $\hat{\delta}$  a função de transição estendida de  $M$ .

Seja  $x \in \Sigma^*$  tal que:

- a)  $x \in L$ .
- b)  $|x| \geq n$ .
- c)  $x = z_1 z_2 z_3$  com  $|z_2| \geq n$ .

Se  $x \in L$  e  $x = z_1 z_2 z_3$  então  $z_1 z_2 z_3 \in L$ .

Se  $z_1 z_2 z_3 \in L$  então  $\hat{\delta}(q_0, z_1 z_2 z_3) \in F$ .

Logo, existem estados  $\{q_1, q_2, q_3\} \subset Q$  tais que:

- i)  $q_1 = \hat{\delta}(q_0, z_1)$ .
- ii)  $q_2 = \hat{\delta}(q_1, z_2)$ .
- iii)  $q_3 = \hat{\delta}(q_2, z_3)$  e  $q_3 \in F$ .

Mas, por construção,  $|z_2| \geq n$ . Ou seja, o número de caracteres de  $z_2$  é maior que o número de estados de  $M$ . Portanto, para que  $M$  vá do estado  $q_1$  para o estado  $q_2$  processando  $z_2$  (tal como descrito no item (ii) acima),  $M$  tem que passar por mais de uma vez em pelo menos um de seus estados. Trata-se do mesmo argumento do “Princípio da Casa dos Pombos” visto na prova do *Lema do Bombeamento*.

Seja  $q_r \in Q$  este estado pelo qual  $M$  tem que passar duas ou mais vezes enquanto processa a cadeia  $z_2$  transitando de  $q_1$  para  $q_2$ .

O *Lema do Bombeamento* garante que existe uma cadeia  $v$ , formada pelos caracteres de  $z_2$  lidos durante a primeira e a segunda passagem de  $M$  por  $q_r$  tal que  $z_2 = uvw$  com  $|v| > 0$  e  $|uv| < n$ .

Seja  $z_i = uv^i w$  para qualquer  $i \in \mathbb{N}$ .

Tendo em vista que  $v$  é um ciclo, temos que  $\hat{\delta}(q_1, z_i) = q_2$  para qualquer  $i \in \mathbb{N}$ .

Portanto, pelas condições (i) a (iii) acima, as cadeias da forma  $z_1 u v^i w z_3$  farão  $M$  transitar de  $q_0$  para  $q_1$ , ao ler  $z_1$ ; de  $q_1$  para  $q_2$ , ao ler  $u v^i w$  e de  $q_2$  para  $q_3$ , ao ler  $z_3$ .

Como  $q_3$  é um estado de aceitação,  $M$  irá aceitar  $z_1 u v^i w z_3$ . □

---

**Exercício 3.3.** Use o Exercício 3.2 para provar que  $L = \{0^i 1^m 2^m \mid i \geq 1, m \geq 1\}$  não é regular.

**Solução:**

**Demonstração.** Por contradição.

Se, por absurdo,  $L$  é regular então  $L$  é aceita por um AFD  $M$ . Seja  $n$  o número de estados de  $M$ .

Seja  $\alpha = 0^n 1^n 2^n = z_1 z_2 z_3$  com  $z_1 = 0^n$ ,  $z_2 = 1^n$  e, portanto,  $z_3 = 2^n$ . Pela definição de  $L$ , temos que  $\alpha \in L$ .

Fazemos  $z_2 = 1^n = uvw$  com  $|v| > 0$  e  $|uv| < n$ , atendendo às condições da extensão do *Lema do Bombeamento*.

Portanto, pela extensão do *Lema do Bombeamento*, para todo  $k \geq 0$  e  $y = uv^k w$  devemos ter  $\beta = z_1 y z_3 \in L$ .

Como  $v$  é uma subcadeia de  $z_2$  a qual é uma cadeia formada apenas por 1's, temos que  $v$  é formado apenas por 1's.

Se  $k = 2$  então  $y = uv^2 w$ . Portanto,  $\beta = z_1 y z_3 = 0^n 1^{n+|v|} 2^n$ .

Porém, como  $|v| > 0$  temos que  $n + |v| \neq n$  e, portanto,  $\beta \notin L$ , o que é uma contradição com a extensão do *Lema do Bombeamento*.

Devido a esta contradição,  $L$  não é regular. □

---

**Exercício 3.4.** Seja  $L$  uma linguagem regular. Quais dos conjuntos seguintes é também regular? Justifique suas respostas.

- a)  $\{a_1 a_2 a_3 \cdots a_{2n-1} \mid a_1 a_2 a_3 \cdots a_{2n} \in L\}$
- b)  $\{a_2 a_1 a_4 a_3 \cdots a_{2n} a_{2n-1} \mid a_1 a_2 \cdots a_{2n} \in L\}$
- c)  $CYCLE(L) = \{x_1 x_2 \mid x_2 x_1 \in L \text{ para cadeias } x_1 \text{ e } x_2\}$
- d)  $MAX(L) = \{x \in L \mid \text{para nenhum } y, \text{ exceto } \epsilon, xy \in L\}$
- e)  $MIN(L) = \{x \in L \mid \text{nenhum prefixo próprio de } x \in L\}$
- f)  $INIT(L) = \{x \mid \text{para algum } yxy \in L\}$
- g)  $L^R = \{x \mid x^R \in L\}$
- h)  $\{x \mid xx^R \in L\}$

**Solução:**

- a) Se  $L$  é uma linguagem regular então  $L' = \{a_1 a_2 a_3 \cdots a_{2n-1} \mid a_1 a_2 a_3 \cdots a_{2n} \in L\}$  é regular também.

**Demonstração.** Por construção.

Se  $L$  é regular então  $L$  é aceita por um AFD  $M = (Q, \Sigma, \delta, q_0, F)$ .

Vamos construir, a partir de  $M$ , o AFD  $M' = (Q', \Sigma', \delta', q'_0, F')$  que aceita  $L'$ .

Se  $a_1 a_2 a_3 \cdots a_{2n} \in L$  então existe um estado  $p \in Q$  tal que:

- i)  $\hat{\delta}(q_0, a_1 a_2 a_3 \cdots a_{2n-1}) = p$  e

ii)  $\hat{\delta}(p, a_{2n}) \in F$

Para reconhecer  $L'$  basta criar um AFD  $M'$  igual a  $M$  com as seguintes alterações:

- a) Os estados de aceitação de  $M$  tornam-se estados comuns.
- b) Os estados que apontam para estados de aceitação em  $M$  (tal como o estado  $p$  acima definido) tornam-se estados de aceitação em  $M'$ .

Se  $L'$  é aceita por um AFD, então  $L'$  é regular. □

- b) Se  $L$  é uma linguagem regular então  $L' = \{a_2a_1a_4a_3 \cdots a_{2n}a_{2n-1} | a_1a_2 \cdots a_{2n} \in L\}$  é regular também.

**Demonstração.** Por construção.

Se  $L$  é regular então  $L$  é aceita por um AFD  $M = (Q, \Sigma, \delta, q_0, F)$ .

Vamos construir, a partir de  $M$ , o AFD  $M' = (Q', \Sigma', \delta', q'_0, F')$  que aceita  $L'$ .

Para cada par  $a_{i-1}a_i$  lido por  $M$ , o AFD  $M'$  deve ler  $a_i a_{i-1}$ . Por exemplo, quando  $M'$  lê  $a_2a_1$ ,  $M$  lê  $a_1a_2$ .

Ou seja,  $M'$  lê a cadeia de entrada aos pares. Primeiro lê o caracter  $a_i$  e o armazena em um de seus estados finitos. Em seguida, ao ler  $a_{i-1}$ ,  $M'$  se comporta como  $M$  processando  $a_{i-1}a_i$ .

Assim, temos:

- i)  $Q' = Q \cup Q \times \Sigma$  pois  $M'$  tem que armazenar em seus estados o primeiro caracter do par lido.
- ii)  $\Sigma' = \Sigma$ .
- iii)  $\delta'(q, a) = [q, a]$  quando  $a \in \Sigma$  é o primeiro caracter do par  $a_i a_{i-1}$  e  $\delta'([q, a], b) = \hat{\delta}(q, ba)$  quando  $b \in \Sigma$  é o segundo caracter do par  $a_i a_{i-1}$ .
- iv)  $q'_0 = q_0$ .
- v)  $F' = F$

Da forma como foi definido acima,  $M'$  aceita  $L'$ . Ou seja:

$$\hat{\delta}'(q, a_2a_1a_4a_3 \dots a_i a_{i-1}) = \hat{\delta}(q, a_1a_2a_3a_4 \dots a_{i-1}a_i) \quad (47)$$

**Demonstração.** Por indução em  $i \in \mathbb{N} | i$  é par.

*Caso base:* Se  $i = 0$  então a cadeia lida é  $\epsilon$ . De fato,

$$\hat{\delta}'(q, \epsilon) = \hat{\delta}(q, \epsilon) = q$$

*Hipótese Indutiva:* Supondo que a equação 47 vale para todo  $j < i$ , temos que existe um estado  $p \in Q$  tal que:

$$\hat{\delta}'(q, a_2a_1a_4a_3 \dots a_{i-2}a_{i-3}) = \hat{\delta}(q, a_1a_2a_3a_4 \dots a_{i-3}a_{i-2}) = p$$

Portanto:

$$\hat{\delta}'(q, a_2a_1a_4a_3 \dots a_{i-2}a_{i-3}a_i a_{i-1}) = \hat{\delta}'(p, a_i a_{i-1})$$

Pela forma como  $M'$  foi construído, este AFD está no estado  $[p, a_i]$  após ter lido o caracter  $a_i$  partindo do estado  $p$ :

$$\hat{\delta}'(p, a_i a_{i-1}) = \hat{\delta}'([p, a_i], a_{i-1})$$

Também por construção,  $M'$  se comporta como  $M$  quando, estado estando no estado  $[p, a_i]$  ele lê o caracter  $a_{i-1}$ :

$$\hat{\delta}'([p, a_i], a_{i-1}) = \hat{\delta}(p, a_{i-1}a_i)$$

Mas, por definição de  $\hat{\delta}$  e de  $p$ :

$$\hat{\delta}(p, a_{i-1}a_i) = \hat{\delta}(q, a_1a_2 \dots a_{i-1}a_i)$$

□

Logo, se  $L'$  é aceita por um AFD então  $L'$  é regular.

□

- c) Se  $L$  é regular então  $CYCLE(L) = \{x_1x_2 | x_2x_1 \in L \text{ para cadeias } x_1 \text{ e } x_2\}$  também é regular.

**Demonstração.** Por construção.

Se  $L$  é regular então  $L$  é aceita por um AFD  $M = (Q, \Sigma, \delta, q_0, F)$ .

Seja  $Q = \{q_1, q_2, \dots, q_n\}$ . Portanto,  $n = |Q|$ .

Se  $w \in L$  então, por definição, existem  $x_1, x_2 \in \Sigma^*$  tal que  $w = x_2x_1$ .

Se  $L$  aceita  $w$  então existem estados  $q_r, q_s \in Q$  tal que  $\hat{\delta}(q_0, x_2) = q_r$  e  $\hat{\delta}(q_r, x_1) = q_s$  com  $q_s \in F$ .

Queremos construir um AFND  $M_c = (Q_c, \Sigma, \delta_c, q_0^c, F_c)$  que reconheça  $CYCLE(L) = \{x_1x_2 | x_2x_1 \in L\}$  usando apenas o que já conhecemos de  $M$ .

Portanto, se  $M$  reconhece  $w = x_2x_1$  queremos que  $M_c$  reconheça  $w_c = x_1x_2$ .

Para que  $M_c$  reconheça  $w_c = x_1x_2$ ,  $M_c$  tem que processar primeiro a cadeia  $x_1$ . Sabemos que  $\hat{\delta}(q_r, x_1) = q_s$  mas não sabemos qual dos  $n$  estados é  $q_r$ .

Assim, criamos um AFND  $M' = (Q', \Sigma, \delta', q_0', F')$  que é uma cópia de  $M$  com algumas alterações:

- (a)  $Q' = Q \cup q_\alpha$ .
- (b)  $q_0' = q_\alpha$ .
- (c)  $F' = \emptyset$ .
- (d) Para todo  $a \in \Sigma$  e para todo  $q_i \in Q'$ :

$$\delta'(q_i, a) = \begin{cases} q & \text{se } q_i = q_\alpha \\ \delta(q_i, a) & \text{caso contrário} \end{cases}$$

Ou seja, o AFND  $M'$  é uma cópia do AFD  $M$  com a adição de um novo estado inicial  $q_\alpha$ , do qual saem transições  $\epsilon$  para os demais  $n$  estados de  $M'$ .

Deste modo, quando  $M'$  recebe o primeiro caracter da cadeia  $x_1$ , as transições  $\epsilon$  enviam este caracter para todos os  $n$  estados  $q_1, q_2, \dots, q_n$ . Um deles é  $q_r$ . Por construção,  $\delta'$  tem igual à  $\delta$  e, como já mostramos que  $\hat{\delta}(q_r, x_1) = q_s$ , temos  $\hat{\delta}'(q_r, x_1) = q_s$ .

Portanto, ao término do processamento de  $x_1$ , certamente uma das instâncias de  $M'$  estará em  $q_s$ .

Após processar  $x_1$  temos que processar  $x_2$ . Sabemos que  $\hat{\delta}(q_0, x_2) = q_r$ .

Portanto, criamos um AFND  $M'' = (Q'', \Sigma, \delta'', q_0'', F'')$  que também é uma cópia de  $M$ .

Além disso, adicionamos transições  $\epsilon$  ligando todos os estados de  $M'$  para  $q_0''$ . Ou seja, ligamos por transições  $\epsilon$  todos os estados de  $M'$  com o estado inicial de  $M''$ . Deste modo, quando  $M'$  concluir o processamento de  $x_1$  estaremos em  $q_0''$  de  $M''$ .

Como  $M''$  é uma cópia de  $M$ , temos que  $\hat{\delta}''(q_0, x_2) = q_r \in F''$ .

Ou seja, ao processarmos  $x_2$  estaremos em um estado de aceitação, reconhecendo  $x_1x_2$  conforme desejado.

Assim, criamos o autômata  $M_c$  tal que:

$$M_c = M' \cup M''$$

Por construção,  $CYCLE(L)$  é reconhecida pelo automata  $M_c$  e, portanto,  $CYCLE(L)$  é regular.

□

- d) Se  $L$  é regular, então  $MAX(L) = \{x \in L | \text{para nenhum } y, \text{ exceto } \epsilon, xy \in L\}$  é regular.

Mais formalmente, podemos definir  $MAX(L)$  como:

$$MAX(L) = \{x \in L | \nexists y \in \Sigma^+ \wedge xy \in L\}$$



**Demonstração.** Por construção.

Se  $L$  é regular então  $L$  é aceita por um AFD  $M = (Q, \Sigma, \delta, q_0, F)$ .

**Definição: Estado Vivo.** Dizemos que um estado  $s_v \in Q$  é um estado *vivo* quando, para pelo menos uma cadeia  $w \in \Sigma^+$  temos  $\hat{\delta}(s_v, w) \in F$ .

Seja  $V \subseteq Q$  o conjunto dos estados vivos de  $M$ .

Construímos  $M' = (Q, \Sigma, \delta, q_0, F - V)$ . Ou seja,  $M'$  é uma cópia de  $M$  com os estados de aceitação alterados de  $F$  para  $F - V$ .

Vamos provar que  $M'$  reconhece  $\text{MAX}(L)$  ou, mais formalmente:

$$x \in \text{MAX}(L) \iff x \in L(M') \quad (48)$$

**Ida:**  $x \in \text{MAX}(L) \Rightarrow x \in L(M')$ .

Se  $x \in \text{MAX}(L)$  então, pela definição de  $\text{MAX}(L)$ :

- i)  $x \in L$ .
- ii)  $\forall y \in \Sigma^+, xy \notin L$ .

De (i) temos que  $\hat{\delta}(q_0, x) = q_f \in F$ .

De (ii) e da definição de  $V$  sabemos que  $q_f \notin V$ .

Logo:  $q_f \in F - V$  e  $F - V$  contém os estados de aceitação de  $M'$ . Portanto,  $x \in L(M')$ .

**Volta:**  $x \in L(M') \Rightarrow x \in \text{MAX}(L)$ .

Se  $x \in L(M')$  então:

- i)  $\hat{\delta}(q_0, x) = q_f \in F$ , pela definição de  $M'$ .
- ii)  $\hat{\delta}(q_0, x) = q_f \notin V$ , pela definição de  $V$ .

De (i) temos que  $x \in L$ .

De (ii) temos que  $\delta(\hat{\delta}(q_0, x), y) \notin F$  para todo  $y \in \Sigma^+$ .

Logo:  $\nexists y \in \Sigma^+ \wedge xy \in L$ . Portanto,  $x \in \text{MAX}(L)$ . □

e) Inicialmente vamos recordar algumas definições.

Sejam  $x, y, z \in \Sigma^*$ .

**Definição: Prefixo.** Dizemos que uma cadeia  $y$  é um *prefixo* de outra cadeia  $x$  se existe uma cadeia  $z$  para a qual  $x = yz$ .

**Definição: Prefixo Próprio.** Dizemos que  $y$  é um *prefixo próprio* de  $x$  se  $y$  é um prefixo de  $x$ , e  $y \neq x$  (ou seja,  $|z| > 0$ ).

Se  $L$  é regular, então  $\text{MIN}(L) = \{x \in L \mid \text{nenhum prefixo próprio de } x \in L\}$  é regular.

**Demonstração.** Por construção.

Se  $L$  é regular então  $L$  é aceita por um AFD  $M = (Q, \Sigma, \delta, q_0, F)$ .

Se  $x = yz \in L$  tal que  $y \in L$  então, por definição,  $y \in \text{MIN}(L)$  e  $x \notin \text{MIN}(L)$  pois  $x$  possui um prefixo próprio em  $L$ .

Quando  $M$  reconhece  $x$ , o automata transita de  $q_0$  por estados não finais até chegar a um estado final, reconhecendo  $y$ . Em seguida, como  $|z| > 0$ ,  $M$  tem que *sair* do estado final em que se encontra para transitar por outros estados até chegar a um estado final que reconhece  $x = yz$ .

Logo, para manter a propriedade de reconhecer  $y$  e remover a propriedade de reconhecer  $x$ , basta remover todos os arcos de  $M$  que saem de estados de aceitação  $q \in F$ .

Assim, construímos  $M' = (Q \cup \{q_\omega\}, \Sigma, \delta', q_0, F - V)$  com:

- (a)  $q_\omega \notin Q$  é um estado armadilha. Uma vez que se chega neste estado, não é possível sair dele não importa o caracter lido na cadeia de entrada.
- (b)  $\delta'$  é idêntica à  $\delta$  exceto que  $\delta'(q, a) = q_\omega$  para todo  $q \in F \cup \{q_\omega\}$  e  $a \in \Sigma$ .

Vamos provar que  $M'$  reconhece  $\text{MIN}(L)$  ou, mais formalmente:

$$x \in \text{MIN}(L) \iff x \in L(M') \quad (49)$$

**Ida:**  $x \in \text{MIN}(L) \Rightarrow x \in L(M')$ .

Se  $x \in \text{MIN}(L)$  então, pela definição de  $\text{MIN}(L)$ :

- i)  $x \in L$ .
- ii)  $x \neq uv$  se  $|v| > 0$  e  $u \in L$ .

Isto equivale a dizer que:

$$x \in \text{MIN}(L) \wedge x = uv \wedge |v| > 0 \Rightarrow u \notin L \quad (50)$$

Ou seja, para qualquer prefixo próprio  $u$  de  $x$  temos  $\hat{\delta}(q_0, u) \notin F$ .

Portanto, como  $\delta'$  é idêntica à  $\delta$  nos estados  $q \in Q - F$ , devemos ter  $\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, x)$  pois, por construção, nenhum prefixo próprio  $u$  de  $x$  passou por um estado final  $q_f \in F$ .

Ou seja,  $x \in L \Rightarrow \hat{\delta}(q_0, x) \in F$  e, desde que  $\hat{\delta}'(q_0, x) = \hat{\delta}(q_0, x)$  temos que  $\hat{\delta}'(q_0, x) \in F$  onde  $F$  é o conjunto dos estados de aceitação de  $M'$ . Logo  $x \in L(M')$

**Volta:**  $x \in L(M') \Rightarrow x \in \text{MIN}(L)$ .

Se  $x \in L(M')$  então  $\hat{\delta}'(q_0, x) \in F$ .

Mas, para ir de  $q_0$  até um estado final  $q_f \in F$ ,  $M'$  não pode ter passado por nenhum estado final intermediário, pois, se o fizesse, teria transitado para o estado não final  $q_\omega \notin F$  pois, por construção,  $\delta'(q, a) = q_\omega$  para todo  $q \in F \cup \{q_\omega\}$  e  $a \in \Sigma$ . Logo, nenhum prefixo próprio  $u$  de  $x$  está em  $L$ .

Além disso, como  $\delta'$  é idêntica à  $\delta$  nos estados  $q \in Q - F$ , devemos ter  $\hat{\delta}(q_0, x) \in F$ . Logo,  $x \in L$ .

Mas, se  $x$  está em  $L$  e nenhum prefixo próprio  $u$  de  $x$  está em  $L$  então  $x \in \text{MIN}(L)$ .  $\square$

- f) Se  $L$  é regular então  $\text{INIT}(L) = \{x \mid \text{para algum } yxy \in L\}$  também é regular.

Note que a definição de  $\text{INIT}(L)$  é praticamente a mesma que a definição de  $\text{MAX}(L) = \{x \in L \mid \text{para nenhum } y, \text{ exceto } \epsilon, xy \in L\}$ . A única diferença é que em  $\text{INIT}(L)$  podemos ter  $y = \epsilon$ .

Portanto a solução é exatamente a mesma vista no item (d) acima, exceto que consideramos estados vivos aqueles que podem ser alcançados por uma cadeia  $y = \epsilon$ .

- g) Se  $L$  é regular então  $L^R$  é regular.

**Demonstração.** Por construção.

Se  $L$  é regular então existe uma expressão regular  $S$  que representa  $L$ . Por definição,  $S$  é definida recursivamente a partir dos caracteres de  $\Sigma^*$  e das operações de concatenação, união e estrela de Kleene.

Vamos mostrar que, a partir dos componentes de  $S$  e usando concatenação, união e estrela de Kleene é possível definir recursivamente uma expressão regular  $S^R$  que equivale a  $L^R$ .

E, se existe uma expressão regular que equivale a  $L^R$ , então  $L^R$  é regular.

Para obter  $S^R$  seguimos estas regras recursivas:

- i) Se  $S = a$  para algum  $a \in \Sigma^*$  então  $S^R = S$  pois todo caracter é igual ao seu reverso.
- ii) Se  $S = S_1 \cup S_2$  então  $S^R = S_2^R \cup S_1^R$ .
- iii) Se  $S = S_1 S_2$  então  $S^R = S_2^R S_1^R$ .
- iv) Se  $S = (S_1)^*$  então  $S^R = (S_1^R)^*$ .

□

h) A linguagem  $L' = \{x|xx^R \in L\}$  não é regular.

**Demonstração.** Pelo teorema de Myhill-Nerode.

Se  $x \in L$  então  $xx^R \in L'$ .

Se  $y \in \Sigma$  então  $y$  distingue  $x^R$  pois se  $y = x$  então  $yx^R \in L'$  e,  $y \neq x$  então  $yx^R \notin L'$ .

Como escolhemos  $y$  genérico, temos que  $L'$  define infinitas classes de equivalência para  $y$ . Logo, pelo teorema de Myhill-Nerode,  $L$  não é regular. □

**Exercício 3.13.** Seja  $h$  um homomorfismo tal que  $h(a) = 01, h(b) = 0$ .

a) Encontre  $h^{-1}(L_1)$  onde  $L_1 = (\mathbf{10} + \mathbf{1})^*$ .

b) Encontre  $h(L_2)$  onde  $L_2 = (\mathbf{a} + \mathbf{b})^*$ .

c) Encontre  $h^{-1}(L_3)$  onde  $L_3$  é o conjunto das cadeias de 0's e 1's com igual número de 0's e 1's.

**Solução:**

a) Se  $h(a) = 01, h(b) = 0$  então  $h^{-1}(01) = a, h^{-1}(0) = b$ . Se  $L_1 = (\mathbf{10} + \mathbf{1})^*$  então  $h^{-1}(L_1) = (\mathbf{10} + \mathbf{1})^*$ .

b) Se  $h(a) = 01, h(b) = 0$  e  $L_2 = (\mathbf{a} + \mathbf{b})^*$  então  $h(L_2) = (\mathbf{01} + \mathbf{0})^*$

c)  $L_3$  é não regular. Portanto,  $h^{-1}(L_3) = \emptyset$ .

**Exercício 3.20.** Ao converter um AFND para um AFD, o número de estados pode crescer substancialmente. Encontre os limites inferior e superior para o aumento máximo do número de estados ao converter, para AFD, um AFND de  $n$  estados.

**Solução:**

Seja  $M_n = (Q, \Sigma, \delta, q_0, F)$  o AFND de  $n$  estados e  $M_d = (Q', \Sigma, \delta', q'_0, F')$  o AFD mínimo correspondente, com  $d$  estados.

Para obter o limite inferior de  $d$  consideramos  $M_n$  com  $q_0 \notin F$  e sem nenhum tipo de ligação entre estados. Isto é, para todo  $a \in \Sigma$  e para todo  $x \in Q$  temos  $\delta(x, a) = \emptyset$ . Deste modo,  $L(M_n) = \emptyset$  e o  $M_d$  mínimo correspondente possui zero estados.

Se considerarmos absurdo um AFD com zero estados, podemos considerar um AFD com um único estado  $q'_0 \notin F'$  e adotar o limite inferior como sendo 1.

Para obter o limite superior, consideramos que cada estado do AFND pode se ligar consigo mesmo e com cada um dos demais  $(n - 1)$  estados.

Para facilitar o raciocínio, vamos considerar que os  $n$  estados do AFND estão numerados como  $1, 2, \dots, n$ .

Se  $n = 1$  então o AFND pode ter no máximo uma ligação consigo mesmo, gerando o estado  $[1]$  no AFD. Também pode ocorrer que, para um certo  $a \in \Sigma$ , não haja arcos saindo do estado 1 o que gera uma ligação do estado  $[1]$  para o estado  $[\emptyset]$  no AFD. Assim,  $Q' = \{[\emptyset], [1]\}$  e  $d = |Q'| = 2^1 = 2$ .

Se  $n = 2$  então o AFD pode ter os estados  $Q' = \{[\emptyset], [1], [2], [12]\}$ . Portanto,  $d = |Q'| = 2^2 = 4$ .

Se  $n = 3$  então o AFD pode ter os estados  $Q' = \{\emptyset, [1], [2], [3], [12], [13], [23], [123]\}$ . Portanto,  $d = |Q'| = 2^3 = 8$ .

Vamos provar, por indução finita em  $n$ , que o limitante superior é  $d = 2^n$ .

**Demonstração.** *Caso base:* Já vimos que a equação vale para  $n = 1$ .

*Hipótese Indutiva:* Supondo que a equação vale para  $n - 1$ , temos que um AFND pode gerar um AFD com até  $d_{n-1} = 2^{n-1}$  estados.

Ao adicionar mais um estado, este pode se conectar consigo mesmo e com cada um dos  $n - 1$  estados já existentes. Cada conexão gera um estado no AFD. Por exemplo, se os  $n - 1$  estados eram 1 e 2 então a adição do estado 3 gera as novas conexões (e correspondentes estados no AFD) 3, 31, 32 e 123. Ou seja, dobra-se o número de estados do AFD.

Portanto, se havia  $d_{n-1} = 2^{n-1}$  estados no AFD, a adição do  $n$ -ésimo estado no AFND, dobra o número total de estados no AFD equivalente. Portanto,  $d_n = 2d_{n-1} = 2 \times 2^{n-1} = 2^n$ .

Supondo a equação válida para  $n - 1$  estados, mostramos que ela continua válida para  $n$  estados.

Portanto, pelo Princípio da Indução Finita, provamos que um AFND de  $n$  estados pode gerar um AFD mínimo com no máximo  $d_n = 2^n$  estados.  $\square$

Em resumo, um AFND de  $n$  estados pode gerar um AFD mínimo que tenha de 0 a  $2^n$  estados.

**Exercício 3.25.** Encontre o AFD mínimo equivalente ao diagrama de transição da Figura 34.

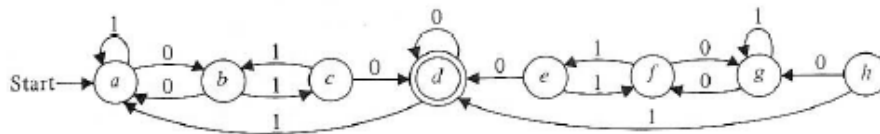


Figura 34 Diagrama de transição para o Exercício 3.25.

**Solução:**

Inicialmente, verificamos que o estado  $h$  não pode ser alcançado a partir do estado inicial (pois não há setas *saindo* deste estado). Portanto, podemos remover o estado  $h$  do AFD a minimizar.

De modo análogo, os estados  $g, f, e$  são sucessivamente removidos pelo mesmo motivo (não há como alcançá-los a partir do estado inicial). Portanto, restam apenas os estados  $a, b, c, d$ .

Em seguida, renumeramos os estados  $a, b, c, d$  para  $q_1, q_2, q_3, q_4$  de modo poder empregar mais facilmente o algoritmo visto em sala de aula<sup>2</sup>.

Aplicando-se o algoritmo, temos:

*Passo 1:* Para cada par  $q_i, q_j$  com  $i < j$  fazer  $D[i, j] = 0$  e  $S[i, j] = \emptyset$ .

*Passo 2:* Para cada par  $q_i, q_j$  com  $i < j$  fazer  $D[i, j] = 1$  se um dos estados ( $q_i$  ou  $q_j$ ) é de aceitação e o outro não é um estado de aceitação.

Portanto:

$$D[1, 4] = D[2, 4] = D[3, 4] = 1$$

<sup>2</sup>Algoritmo 5.7.2 na pág. 179 do livro SUDKAMP, Thomas A., *Languages and Machines. An Introduction to The Theory of Computer Science*. 3ed. 2005. ISBN 9780321322210

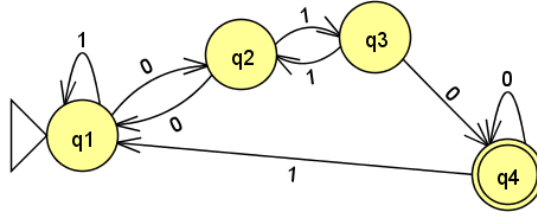


Figura 35 Diagrama de transição para o Exercício 3.25 após os estados  $e, f, g, h$  terem sido removidos e os estados  $a, b, c, d$  terem sido renomeados para  $q_1, q_2, q_3, q_4$ .

*Passo 3:* Para cada par  $q_i, q_j$  com  $i < j$  em que  $D[i, j] = 0$  executamos o passo 3 tal como resumido na tabela abaixo.

Índice	Ação	Razão
[1, 2]	$S[1, 3] = \{[1, 2]\}$ $S[1, 2] = \{[1, 2]\}$	
[1, 3]	$D[2, 3] = 1$ $D[1, 3] = 1$	Distintos por 0 Chamando $DIST(1, 3)$
[2, 3]	$D[2, 3] = 1$	Distintos por 0

Portanto, os estados são distintos e o AFD da Figura 35 é mínimo.

**Exercício 1.** Seja  $G$  a gramática:

$$\begin{aligned} S &\rightarrow SAB | \epsilon \\ A &\rightarrow aA | a \\ B &\rightarrow bB | \epsilon \end{aligned}$$

- Dê a derivação mais à esquerda de 'abbaab'.
- Dê duas derivações mais à esquerda de 'aa'.
- Construa a árvore de derivações para o item (b).
- Dê uma expressão regular para  $L(G)$ .

**Solução:**

- Derivação mais à esquerda de 'abbaab':

$$\begin{aligned} S &\Rightarrow SAB & (S \rightarrow SAB) \\ SAB &\Rightarrow SABAB & (S \rightarrow SAB) \\ SABAB &\Rightarrow ABAB & (S \rightarrow \epsilon) \\ ABAB &\Rightarrow aBAB & (A \rightarrow a) \\ aBAB &\Rightarrow abBAB & (B \rightarrow bB) \\ abBAB &\Rightarrow abbBAB & (B \rightarrow bB) \\ abbBAB &\Rightarrow abbAB & (B \rightarrow \epsilon) \\ abbAB &\Rightarrow abbaAB & (A \rightarrow aA) \\ abbaAB &\Rightarrow abbaaB & (A \rightarrow a) \\ abbaaB &\Rightarrow abbaabB & (B \rightarrow bB) \\ abbaabB &\Rightarrow abbaab & (B \rightarrow \epsilon) \end{aligned}$$

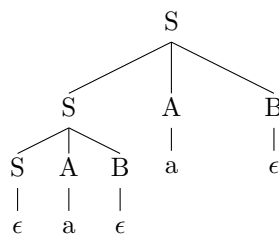
b) Primeira derivação mais à esquerda de 'aa':

$$\begin{aligned}
 S &\Rightarrow SAB & (S \rightarrow SAB) \\
 SAB &\Rightarrow SABAB & (S \rightarrow SAB) \\
 SABAB &\Rightarrow ABAB & (S \rightarrow \epsilon) \\
 ABAB &\Rightarrow aBAB & (A \rightarrow a) \\
 aBAB &\Rightarrow aAB & (B \rightarrow \epsilon) \\
 aAB &\Rightarrow aaB & (A \rightarrow a) \\
 aaB &\Rightarrow aa & (B \rightarrow \epsilon)
 \end{aligned}$$

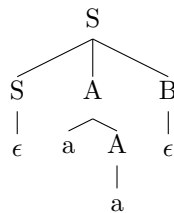
Segunda derivação mais à esquerda de 'aa'.

$$\begin{aligned}
 S &\Rightarrow SAB & (S \rightarrow SAB) \\
 SAB &\Rightarrow AB & (S \rightarrow \epsilon) \\
 AB &\Rightarrow aAB & (A \rightarrow aA) \\
 aAB &\Rightarrow aaB & (A \rightarrow a) \\
 aaB &\Rightarrow aa & (B \rightarrow \epsilon)
 \end{aligned}$$

c) Árvore de derivações para a primeira derivação mais à esquerda de 'aa':



Árvore de derivações para a segunda derivação mais à esquerda de 'aa':



d) Iniciamos com uma expressão regular mais simples e vamos generalizando até encontrar a expressão regular equivalente à gramática.

$S \rightarrow \epsilon$	$\epsilon$
$S \rightarrow SAB$ $S \rightarrow \epsilon$ $A \rightarrow aA$ (várias vezes) $A \rightarrow a$ $B \rightarrow \epsilon$	$\mathbf{a^*}$
$S \rightarrow SAB$ $S \rightarrow \epsilon$ $A \rightarrow a$ $B \rightarrow bB$ (várias vezes) $B \rightarrow \epsilon$	$\mathbf{a^* + ab^*}$
$S \rightarrow SAB$ (várias vezes) $S \rightarrow \epsilon$ $A \rightarrow aA$ (várias vezes) $A \rightarrow a$ $B \rightarrow bB$ (várias vezes) $B \rightarrow \epsilon$	$\mathbf{a^* + ab^* + (a^*b^*)^*}$

Portanto, a expressão regular para  $L(G)$  é:

$$L(G) = \mathbf{a(a + b)^* + \epsilon}$$

que equivale a:

$$L(G) = \{(a^n b^m)^k | n > m \geq 0 \wedge k \geq 0\}$$

**Exercício 2.** Para cada uma das GLC's abaixo, defina a linguagem gerada:

$$a) \quad \begin{array}{l} S \rightarrow aaSB | \epsilon \\ B \rightarrow bB \mid b \end{array}$$

$$b) \quad \begin{array}{l} S \rightarrow aSbb | A \\ A \rightarrow cA \mid c \end{array}$$

$$c) \quad \begin{array}{l} S \rightarrow abSdc | A \\ A \rightarrow cdAba \mid \epsilon \end{array}$$

$$d) \quad \begin{array}{l} S \rightarrow aSb | A \\ A \rightarrow cAd \mid cBd \\ B \rightarrow aBb \mid ab \end{array}$$

$$e) \quad \begin{array}{l} S \rightarrow aSb | AB \\ B \rightarrow bb \mid b \end{array}$$

**Solução:**

Em todas as definições abaixo  $\{i, j, k\} \subset \mathbb{N}$ .

$$a) \quad L(G) = \{a^{2i}b^j | i \geq 1 \wedge j \geq 1\}.$$

$$b) \quad L(G) = \{a^i c^j b^{2i} | i \geq 0 \wedge j \geq 1\}.$$

$$c) \quad L(G) = \{(ab)^i (cd)^j (ba)^j (dc)^i | i \geq 0 \wedge j \geq 1\}.$$

$$d) \quad L(G) = \{a^i c^j (ab)^k d^j b^i | i \geq 0 \wedge j \geq 1 \wedge k \geq 1\}.$$

$$e) \quad L(G) = \{a^i b^j | i > 0 \wedge j \geq i\}$$

**Exercício 3.** Para cada uma das seguintes Gramáticas Regulares, encontra uma expressão regular para a linguagem gerada.

$$a) \quad \begin{array}{l} S \rightarrow aA | \epsilon \\ A \rightarrow aA \mid bA \mid b \end{array}$$

$$b) \quad \begin{array}{l} S \rightarrow aA \\ A \rightarrow aA \mid bB \\ B \rightarrow bB \mid \epsilon \end{array}$$

$$c) \quad \begin{array}{l} S \rightarrow aS \mid bA \\ A \rightarrow bB \\ B \rightarrow aB \mid \epsilon \end{array}$$

$$d) \begin{array}{l} S \rightarrow aS \mid bA \mid \epsilon \\ A \rightarrow aA \mid bS \end{array}$$

### Solução:

Para cada gramática, mostramos algumas derivações e a expressão regular equivalente.

Iniciamos com uma expressão regular mais simples e vamos generalizando até encontrar a expressão regular equivalente à gramática.

a) A expressão regular equivalente à gramática (a) é:

$$\mathbf{a^+b^+ + \epsilon}$$

$S \rightarrow \epsilon$	$\epsilon$
$S \rightarrow aA$ $A \rightarrow aA$ $A \rightarrow b$	$\mathbf{aa^*b + \epsilon = a^+b + \epsilon}$
$S \rightarrow aA$ $A \rightarrow bA$ $A \rightarrow b$	$\mathbf{a^+b + \epsilon + ab^*b}$

b) A expressão regular equivalente à gramática (b) é:

$$\mathbf{a^+b^+}$$

$S \rightarrow aA$ $A \rightarrow bB$ $B \rightarrow \epsilon$	$\mathbf{ab}$
$S \rightarrow aA$ $A \rightarrow aA$ $A \rightarrow bB$ $B \rightarrow \epsilon$	$\mathbf{aa^*b}$
$S \rightarrow aA$ $A \rightarrow bB$ $B \rightarrow bB$ $B \rightarrow \epsilon$	$\mathbf{aa^*b + abb^*}$

c) A expressão regular equivalente à gramática (c) é:

$$\mathbf{a^+bba^*}$$

$S \rightarrow aS$ $S \rightarrow bA$ $A \rightarrow bB$ $B \rightarrow \epsilon$	$\mathbf{aa^*bb}$
$S \rightarrow aS$ $S \rightarrow bA$ $A \rightarrow bB$ $B \rightarrow aB$ $B \rightarrow \epsilon$	$\mathbf{aa^*bb + aa^*bba^*}$

d) A expressão regular equivalente à gramática (d) é:

$$\mathbf{a^* + bbb^* + (ba^*b)^*}$$



$S \rightarrow \epsilon$	$\epsilon$
$S \rightarrow aS$ $S \rightarrow \epsilon$	$\mathbf{a}^*$
$S \rightarrow bA$ $A \rightarrow bS$ $S \rightarrow \epsilon$	$\mathbf{a}^* + \mathbf{bbb}^*$
$S \rightarrow bA$ $A \rightarrow aA$ $A \rightarrow bS$ $S \rightarrow \epsilon$	$\mathbf{a}^* + \mathbf{bbb}^* + (\mathbf{ba}^*\mathbf{b})^*$

**Exercício 4.** *Seja  $G$  a gramática:*

$$\begin{aligned} S &\rightarrow aSb \mid B \\ B &\rightarrow bB \mid b \end{aligned}$$

*Prove que  $L(G) = \{a^n b^m \mid 0 \leq n < m\}$*

**Solução:**

**Demonstração.** Por construção.

Após aplicar  $n \geq 0$  vezes a produção  $S \rightarrow aSb$  obtemos uma cadeia na forma  $a^n S b^n$ .

Porém, como  $S$  não é terminal, temos que aplicar pelo menos uma vez a produção  $S \rightarrow B$  que resulta na cadeia  $a^n B b^n$ .

Como  $B$  não é terminal, podemos aplicar  $k \geq 0$  vezes a produção  $B \rightarrow bB$  obtendo a cadeia  $a^n b^k B b^n$ .

Porém, como  $B$  não é terminal, temos que aplicar pelo menos uma vez a produção  $B \rightarrow b$  que resulta na cadeia  $a^n b^{k+1} b^n$  que é igual à cadeia  $a^n b^{k+1+n}$ .

Fazendo  $m = k + 1 + n$  temos que  $L(G) = a^n b^m$ .

Como  $k \geq 0$  temos que  $m = k + 1 + n > n \geq 0$ .

Logo:  $L(G) = a^n b^m$  com  $m > n \geq 0$ . □

**Exercício 5.** *Determine uma GLC sem símbolos inúteis equivalente a:*

$$\begin{aligned} S &\rightarrow AB \mid CA \\ A &\rightarrow a \\ B &\rightarrow BC \mid AB \\ C &\rightarrow aB \mid b \end{aligned}$$

**Solução:**

Seja  $G = (V, T, P, S)$  uma gramática.

Um símbolo  $X$  é útil se:

- i)  $X \xRightarrow{*} w \in T^*$ , ou seja, o símbolo  $X$  gera uma cadeia formada por símbolos terminais.

ii)  $S \xRightarrow{*} \alpha X \beta$ , ou seja, é possível chegar ao símbolo  $X$  a partir de produções originadas a partir do símbolo inicial  $S$ .

Analisando a gramática, vemos que o símbolo  $B$  não gera símbolos terminais. Logo,  $B$  não é útil. Portanto podemos remover todas as produções que utilizam esta variável. Assim, a gramática pode ser reescrita como:

$$\begin{aligned} S &\rightarrow cA \\ A &\rightarrow a \\ C &\rightarrow b \end{aligned}$$

Agora vemos que  $C$  é não alcançável a partir de  $S$  e que a variável  $A$  apenas produz um único símbolo terminal  $a$ .

Portanto, a gramática pode ser reescrita como:

$$S \rightarrow ca$$