

6.8(b) apresenta o autômato finito determinístico em sua versão final.

## 6.4 Minimização de Autômatos

Neste momento, parece sugestível que não há um algoritmo que permita a construção de um autômato qualquer que implemente a linguagem  $L$  a ser representada. A construção de autômatos envolve um processo eminentemente criativo, que tende a ser facilitado com a prática e o estudo. Durante o processo de criação, o projetista do autômato, cria um arranjo de estados e transições, sem que necessariamente esteja preocupado com a quantidade dos mesmos. Em geral, problemas desta natureza, objetivando uma otimização, são deixados para uma etapa posterior.

Na seção 6.3 foi apresentado o processo de conversão de um NFA em um DFA. Neste processo, observa-se que também não há uma preocupação formal com a redução da quantidade de estados. Existe apenas um ensaio de otimização quando se efetua a verificação de estados inatingíveis e posteriormente realiza-se a eliminação.

Poder calcular o autômato mínimo é uma questão importante na construção de ferramentas baseadas em autômatos finitos tais como: protocolos de comunicação, processamento de texto, análise de imagens, lingüística computacional, entre outras. Isto reduz a complexidade da engenharia do software, e em última medida, torna o processamento da aplicação mais leve, e seu custo financeiro, menor.

Um autômato com  $m$  estados e que aceite uma linguagem  $L$  é dito mínimo quando, para todo DFA de  $n$  estados que aceite a mesma linguagem  $L$ , se a relação entre estas quantidades de estados é dada por  $m < n$ . Dois estados  $q$  e  $q'$  são ditos equivalentes ( $\equiv$ ) se  $\forall \sigma, \delta(q, \sigma) \in F \Leftrightarrow \delta(q', \sigma) \in F$ . Alternativamente, diz-se que dois estados são distintos ( $\not\equiv$ ) se  $\exists \sigma, \delta(q, \sigma) \in F \wedge \delta(q', \sigma) \notin F$ , ou vice-versa.

Observe o autômato da Figura 6.9(a), cuja linguagem aceita é dada por  $(a, b)(a, b)^*$ . Analisando este autômato, percebe-se que a transição de  $q_0$  para  $q_2$  aponta para a construção de strings do tipo  $a(a, b)^*$ . Ainda nesta mesma figura, as transições que passam pelo estado  $q_1$  permitem a construção de strings do tipo  $b(a, b)^*$  de um modo pouco convencional. Isto ocorre porque a passagem por  $q_1$

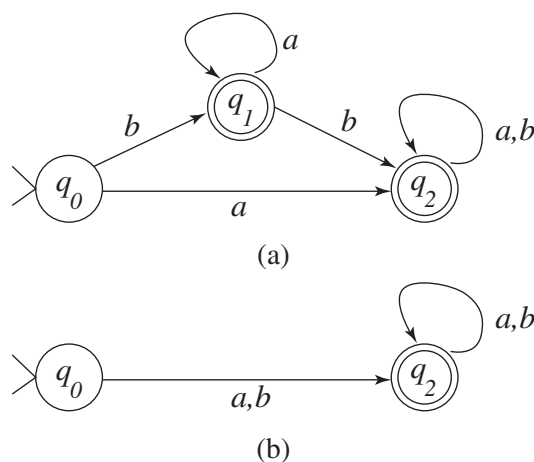


Figura 6.9: Exemplo de minimização: (a) Autômato finito determinístico não minimizado; (b) DFA minimizado.

força a existência de um símbolo  $b$  no início da string. Depois disso, é possível fazer um  $a^*b$ , e depois em  $q_2$  se pode combinar  $a$  ou  $b$  como se bem entender. Note que existem dois comportamentos distintos nesta passagem: (1) quando são repetidos os vários símbolos  $a$ ; (2) quando não se usa o símbolo  $a$  e efetua-se a transição de  $q_1$  para  $q_2$ , através do símbolo  $b$ . O efeito poderia ser reproduzido se fosse criada a transição  $\delta(q_0, b) = q_2$ , combinada com a transição já existente de  $q_0$  para  $q_2$ . Desta forma, toda a parte de cima do autômato poderia ser descartada, produzindo um autômato menor, conforme a Figura 6.9(b).

A minimização descrita para a Figura 6.9 torna-se proibitiva quando os autômatos a serem minimizados ficam maiores, pois a tarefa de determinação do que minimizar passa por um processo de escolha às vezes pouco sistemático, e dependente da experiência do projetista. O processo informal de minimização pode ser descrito em dois passos: (1) identificar todos os estados não equivalentes, chamados de distintos; (2) combinar os estados restantes, os estados equivalentes, respeitando as transições necessárias para que a linguagem  $L$  seja preservada.

Sob esta ótica, dois estados equivalentes serão aqueles que executam tarefas semelhantes. Mais especificamente, dois estados  $q$  e  $q'$  são distintos se, e somente se, estando no estado  $q$  e recebendo um símbolo  $\sigma$ , o DFA efetua a transição para um estado de aceitação (estado final), enquanto que estando no estado  $q'$ , e recebendo

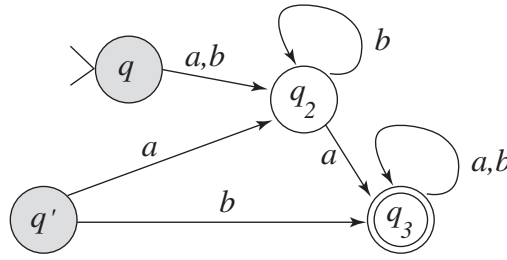


Figura 6.10: Exemplo de não equivalência entre estados.

o mesmo símbolo  $\sigma$ , o DFA efetua a transição para um estado de não aceitação. Observe o exemplo da Figura 6.10, o símbolo  $a$  não permite fazer a distinção entre os estados  $q$  e  $q'$  pois ambos conduzem o autômato para um estado de não aceitação, isto é, que não pertence a  $F$ . Entretanto, o recebimento de um símbolo  $b$ , estando no estado  $q$  conduz para o estado de não aceitação  $q_2$ , enquanto que a partir do estado  $q'$  atinge-se o estado final  $q_3$ . Esta distinção entre os estados de chegada torna os estados  $q$  e  $q'$  distintos.

Outro conceito importante é o de particionamento de um conjunto, que permite dividir um conjunto em subconjuntos sem sobreposição. Suponha um conjunto de estados  $k = \{q_0, q_1, q_2, q_3, q_4, q_5\}$ . Uma partição hipotética deste conjunto pode ser dada por  $P_i = \{\{q_0, q_1\}, \{q_2, q_3, q_4\}, \{q_5\}\}$ . Esta partição, por sua vez, pode ser operada por um operador de refinamento que divide os subconjuntos de  $P_i$  em novos subconjuntos, também sem sobreposição, como por exemplo  $P_{i+1} = \{\{q_0\}, \{q_1\}, \{q_2, q_3, q_4\}, \{q_5\}\}$ . Observe que não é possível efetuar o particionamento de um conjunto unitário, pois o conjunto  $\{\}$  não é considerado como um grupo de partição.

O particionamento inicial de todos conjunto de estados é dado por  $P_0 = \{k - F, F\}$ . Desta forma, na Figura 6.11(a) são apresentados cinco estado, cujo particionamento inicial é dado por  $P_0 = \{\{q_0, q_1, q_4\}, \{q_2, q_3\}\}$ , onde  $G_1 = \{q_0, q_1, q_4\}$

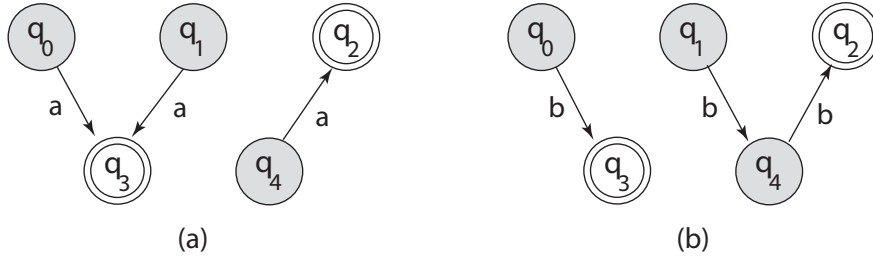


Figura 6.11: Exemplo de particionamento.

e  $G_2 = \{q_2, q_3\}$ . Considere o grupo  $G_1 = \{q_0, q_1, q_4\}$  sendo submetido a entrada  $a$ :

$$\delta(q_0, a) = q_3 \in G_2$$

$$\delta(q_1, a) = q_3 \in G_2$$

$$\delta(q_4, a) = q_2 \in G_2$$

para a entrada  $a$ , todos os estados pertencentes a  $G_1$  são encaminhados para outros estados pertencentes a um mesmo grupo ( $G_2$  no caso), e por isso não é possível fazer uma distinção entre eles. Entretanto, considerando a Figura 6.11(b) tem-se:

$$\delta(q_0, b) = q_3 \in G_2$$

$$\delta(q_1, b) = q_4 \in G_1$$

$$\delta(q_4, b) = q_2 \in G_2$$

o estado  $q_1$  é distinto dos estados  $q_0$  e  $q_4$  já que, cada qual são conduzidos a grupos distintos. Desta forma a partição  $P_0$  pode ser refinada para  $P_1 = \{\{q_0, q_4\}, \{q_1\}, \{q_2, q_3\}\}$ .

Existem duas grandes famílias de algoritmos para minimização. A primeira utiliza uma série de refinamentos das partições do conjunto de estados, e a segunda utiliza uma seqüência de fusões de estados. Os algoritmos de Hopcroft e de Moore pertencem a primeira família, enquanto que o algoritmo de Revuz pertence a segunda. Berstel *et alli* [86] fornecem uma excelente comparação entre os algoritmos de Hopcroft e Moore.

A seguir é descrito o algoritmo de Hopcroft, que é um dos algoritmos mais velozes para efetuar minimização ( $O(m n \log n)$ ,  $m$  é o número de símbolos do alfabeto e  $n$  a quantidade de estados) [1] em casos gerais. Este algoritmo foi estendido por

Béal e Crochemore [87], Valmari e Lehtinen [88], e otimizado para casos específicos. A seguir é apresentado o algoritmo 6.1 conceitualmente na sua versão original, mas adaptado para tornar-se mais legível.

**Data:** DFA  $M = \langle k, \Sigma, \delta, s, F \rangle$  a ser minimizado

**Result:** Particionamento  $P$  correspondente a minimização

```

/* Cria a partição inicial                                     */
1   $P = \{F, k - F\}$ ;
2  while true do
    /* Início do refinamento da partição  $P_{novo}$              */
3     $P_{novo} = P$  ;
4    foreach grupo  $G$  em  $P_{novo}$  do
        // Quebrar  $G$  em subgrupos
5        foreach  $\sigma \in \Sigma$  do
6            Determinar a que grupo pertence  $\delta(q_i^G, \sigma)$  ;
7            Reagrupar os estados  $q_i^G$  associados ao mesmo grupo de
            chegada ;
8            Dar novas etiquetas para os grupos no reagrupamento
             $(G', G'', \dots)$ ;
9            Remover o grupo  $G$  do conjunto  $P_{novo}$  ;
10           Adicionar os novos grupos  $G', G'', \dots$  ao conjunto  $P_{novo}$  e
            interromper o loop de  $\sigma \in \Sigma$ ;
11        end
12    end
    /* Fim do refinamento da partição  $P_{novo}$                  */
13    if  $P_{novo} == P$  then
14        break;
15    end
16     $P = P_{novo}$ ;
17 end

```

**Algoritmo 6.1:** Minimização de um DFA  $M = \langle k, \Sigma, \delta, s, F \rangle$ , segundo Hopcroft [1].

A proposta do algoritmo de minimização 6.1 é realizar sucessivos refinamen-

tos do particionamento  $P$  até o processo de refinamento não efetue mais nenhuma modificação do conjunto particionado. Quando isto ocorre o algoritmo é finalizado (linhas 13 e 14).

Ao término do algoritmo, e caso seja cabível a minimização, o conjunto  $P$  será composto por novos grupos. Cada um destes grupos corresponderá a um novo estado do autômato minimizado, sendo que o grupo que contiver um estado inicial corresponderá ao estado inicial do novo autômato, e os grupos que contiverem estados finais serão convertidos em estados finais no novo autômato. Dentro de cada grupo, deve-se escolher um dos estados para ser mantido, enquanto os restantes são descartados. As arestas de entrada e saída deste estado devem ser preservadas, eventualmente sendo necessário adaptá-las para um laço de loop. Em seguida devem ser removidos quais estados inatingíveis a partir do estado inicial.

Para exemplificar o funcionamento do algoritmo, retome o exemplo da Figura 6.9(a). O passo a passo do algoritmo é apresentado a seguir:

linha	:	operação
1	:	$P = \{\{q_1, q_2\}, \{q_0\}\}$
3	:	$P_{novo} = \left\{ \underbrace{\{q_1, q_2\}}_{G_1}, \underbrace{\{q_0\}}_{G_2} \right\}$
4	:	$G_1 = \{q_1, q_2\}$
5	:	$\sigma = a$
6	:	$\delta(q_1, a) = q_1 \in G_1$
	:	$\delta(q_2, a) = q_2 \in G_1$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer
5	:	$\sigma = b$
6	:	$\delta(q_1, b) = q_2 \in G_1$
	:	$\delta(q_2, b) = q_2 \in G_1$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer

Neste momento chega ao fim o loop referente a linha 5 do algoritmo. O próximo

passo a ser executado é a nova iteração do loop da linha 4, onde toma-se como novo grupo o  $G_2 = \{q_0\}$ . Observe que o trecho de código entre as linhas 3 e 12 se propõe a fazer um refinamento da partição  $P_{novo}$ . Isto, em última análise, significa particionar o grupo  $G_2$ . Entretanto, este mesmo grupo  $G_2$  é um conjunto unitário, e não é possível mais particioná-lo, sugerindo que as próximas operações são desnecessárias. Contudo, por completude do exemplo proposto, e para demonstrar a insensibilidade desta situação por parte do algoritmo, será dada continuidade ao passo a passo do algoritmo.

linha	:	operação
4	:	$G_2 = \{q_0\}$
5	:	$\sigma = a$
6	:	$\delta(q_0, a) = q_2 \in G_1$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer
5	:	$\sigma = b$
6	:	$\delta(q_0, b) = q_1 \in G_1$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer

De fato, esta última iteração não tinha como acrescentar nenhuma modificação ao  $P_{novo}$ . Por fim, o algoritmo verifica que de fato  $P_{novo} == P$  (linha 13) e interrompe sua execução. Como resultado, é obtido o conjunto de particionamento  $P = \{\{q_1, q_2\}, \{q_0\}\}$ , indicando que  $q_1$  e  $q_2$  são equivalentes.

Dando continuidade a etapa posterior ao algoritmo, observa-se que não há estado a ser escolhido no grupo  $G_2 = \{q_0\}$  porque a única escolha possível, é o único elemento do grupo, isto é,  $q_0$ . Já no grupo  $G_1 = \{q_1, q_2\}$  deve-se escolher um dos estados para ser mantido no autômato minimizado, enquanto os restantes devem ser descartados. Neste caso, optou-se por escolher, aleatoriamente, o estado  $q_0$  e descartar o estado  $q_1$ . As transições que precisam ser adaptadas são aquelas que saem de  $q_0$  e  $q_1$  e seguem para  $q_2$ . Por fim, ao analisar o autômato restante, percebe-se que não existem estados inatigíveis, resultando na minimização da Figura

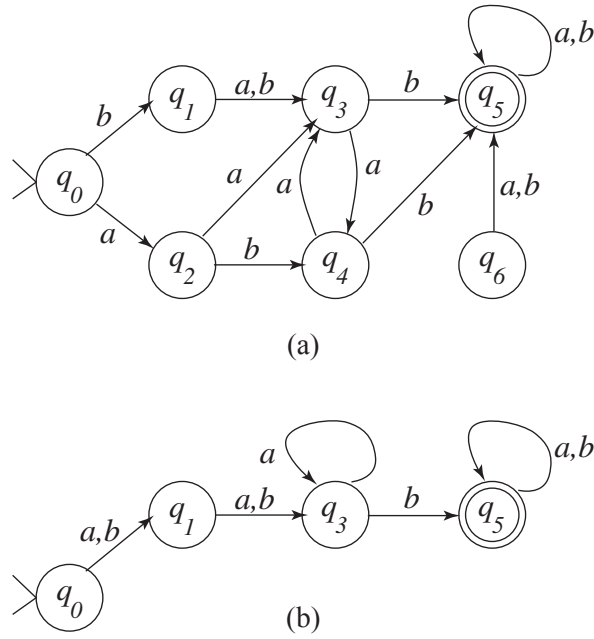


Figura 6.12: Minimização de DFA: (a) autômato original; (b) autômato minimizado.

6.9(b).

O objetivo do exemplo apresentado foi familiarizar o leitor com a mecânica do funcionamento do Algoritmo 6.1. A Figura 6.12(a) apresenta um outro DFA que será usado como ilustração adicional do processo de minimização. Trata-se de um exemplo um pouco mais longo, onde irão surgir algumas situações que ficaram escondidas no exemplo anterior. Neste sentido, um primeiro momento da execução do algoritmo pode ser descrito como se segue:



linha : operação

$$1 : P = \{\{q_0, q_1, q_2, q_3, q_4, q_6\}, \{q_5\}\}$$

$$3 : P_{novo} = \left\{ \underbrace{\{q_0, q_1, q_2, q_3, q_4, q_6\}}_{G_1}, \underbrace{\{q_5\}}_{G_2} \right\}$$

$$4 : G_1 = \{q_0, q_1, q_2, q_3, q_4, q_6\}$$

$$5 : \sigma = a$$

$$6 : \delta(q_0, a) = q_2 \in G_1$$

$$\delta(q_1, a) = q_3 \in G_1$$

$$\delta(q_2, a) = q_3 \in G_1$$

$$\delta(q_3, a) = q_4 \in G_1$$

$$\delta(q_4, a) = q_3 \in G_1$$

$$\delta(q_6, a) = q_5 \in G_2$$

$$7, 8 : G_3 = \{q_0, q_1, q_2, q_3, q_4\}, G_4 = \{q_6\}$$

$$9, 10 : P_{novo} = \left\{ \underbrace{\{q_0, q_1, q_2, q_3, q_4\}}_{G_3}, \underbrace{\{q_6\}}_{G_4}, \underbrace{q_5}_{G_2} \right\}$$

Observe que nesta etapa do algoritmo foi necessário substituir o grupo  $G_1$  pelo grupos  $G_3$  e  $G_4$ . Isso provocou uma alteração no conjunto de particionamento  $P_{novo}$ , o que vai obrigar que o loop da linha 4 recomece a varredura sobre  $P_{novo}$ . Deste

modo, o próximo grupo a ser testado será  $G_3$ .

linha : operação

$$4 : G_3 = \{q_0, q_1, q_2, q_3, q_4\}$$

$$5 : \sigma = a$$

$$6 : \delta(q_0, a) = q_2 \in G_3$$

$$\delta(q_1, a) = q_3 \in G_3$$

$$\delta(q_2, a) = q_3 \in G_3$$

$$\delta(q_3, a) = q_4 \in G_3$$

$$\delta(q_4, a) = q_3 \in G_3$$

$$7 : \text{Não há reagrupamento a ser feito}$$

$$8, 9, 10 : \text{Nada a fazer}$$

$$5 : \sigma = b$$

$$6 : \delta(q_0, b) = q_1 \in G_3$$

$$\delta(q_1, b) = q_3 \in G_3$$

$$\delta(q_2, b) = q_4 \in G_3$$

$$\delta(q_3, b) = q_5 \in G_2$$

$$\delta(q_4, b) = q_5 \in G_2$$

$$7, 8 : G_5 = \{q_0, q_1, q_2\}, G_6 = \{q_3, q_4\}$$

$$9, 10 : P_{\text{novo}} = \left\{ \underbrace{\{q_0, q_1, q_2\}}_{G_5}, \underbrace{\{q_3, q_4\}}_{G_6}, \underbrace{\{q_6\}}_{G_4}, \underbrace{\{q_5\}}_{G_2} \right\}$$

Novamente houve uma alteração no conjunto de particionamento  $P_{\text{novo}}$ , o que vai

provocar um recomeço da varredura sobre  $P_{novo}$ .

linha : operação

$$4 : G_5 = \{q_0, q_1, q_2\}$$

$$5 : \sigma = a$$

$$6 : \delta(q_0, a) = q_2 \in G_5$$

$$\delta(q_1, a) = q_3 \in G_6$$

$$\delta(q_2, a) = q_3 \in G_6$$

$$7, 8 : G_7 = \{q_0\}, G_8 = \{q_1, q_2\}$$

$$9, 10 : P_{novo} = \left\{ \underbrace{\{q_0\}}_{G_7}, \underbrace{\{q_1, q_2\}}_{G_8}, \underbrace{\{q_3, q_4\}}_{G_6}, \underbrace{\{q_6\}}_{G_4}, \underbrace{\{q_5\}}_{G_2} \right\}$$

Nesta iteração, ocorre outra alteração no conjunto de particionamento  $P_{novo}$ , fazendo com que o loop da linha 4 recomece uma nova varredura sobre  $P_{novo}$ .

linha : operação

$$4 : G_7 = \{q_0\}$$

$$5 : \sigma = a$$

$$6 : \delta(q_0, a) = q_2 \in G_8$$

$$7 : \text{Não há reagrupamento a ser feito}$$

$$8, 9, 10 : \text{Nada a fazer}$$

$$5 : \sigma = b$$

$$6 : \delta(q_0, b) = q_1 \in G_8$$

$$7 : \text{Não há reagrupamento a ser feito}$$

$$8, 9, 10 : \text{Nada a fazer}$$

O grupo a ser avaliado foi o  $G_7$ . Nessa avaliação já era esperado que nada acontecesse

pois o conjunto é unitário. Em seguida, o próximo grupo a ser testado é o  $G_8$ .

linha	: operação
4	: $G_8 = \{q_1, q_2\}$
5	: $\sigma = a$
6	: $\delta(q_1, a) = q_3 \in G_6$
	: $\delta(q_2, a) = q_3 \in G_6$
7	: Não há reagrupamento a ser feito
8, 9, 10	: Nada a fazer
5	: $\sigma = b$
6	: $\delta(q_1, b) = q_3 \in G_6$
	: $\delta(q_2, b) = q_4 \in G_6$
7	: Não há reagrupamento a ser feito
8, 9, 10	: Nada a fazer

O grupo  $G_8$  era candidato a sofrer alteração, porém isto não aconteceu. Em seguida, o terceiro grupo a ser avaliado é o  $G_6$ .

linha	: operação
4	: $G_6 = \{q_3, q_4\}$
5	: $\sigma = a$
6	: $\delta(q_3, a) = q_4 \in G_6$
	: $\delta(q_4, a) = q_3 \in G_6$
7	: Não há reagrupamento a ser feito
8, 9, 10	: Nada a fazer
5	: $\sigma = b$
6	: $\delta(q_3, b) = q_5 \in G_2$
	: $\delta(q_4, b) = q_5 \in G_2$
7	: Não há reagrupamento a ser feito
8, 9, 10	: Nada a fazer

A seguir o grupo  $G_4$ , tal como aconteceu anteriormente, é um conjunto unitário, e

portanto é esperado que não haja alteração.

linha	:	operação
4	:	$G_4 = \{q_0\}$
5	:	$\sigma = a$
6	:	$\delta(q_0, a) = q_2 \in G_8$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer
5	:	$\sigma = b$
6	:	$\delta(q_0, b) = q_1 \in G_8$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer

Analogamente, o grupo  $G_2$  também é um conjunto unitário e portanto não haverá alteração.

linha	:	operação
4	:	$G_2 = \{q_5\}$
5	:	$\sigma = a$
6	:	$\delta(q_5, a) = q_5 \in G_2$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer
5	:	$\sigma = b$
6	:	$\delta(q_5, b) = q_5 \in G_2$
7	:	Não há reagrupamento a ser feito
8, 9, 10	:	Nada a fazer

Neste momento, o algoritmo segue um fluxo de execução, que até então não havia sido percorrido. Trata-se da verificação da condição de parada do algoritmo.

linha	:	operação
13	:	$P_{novo}$ é diferente de $P$
14	:	$P = \{\{q_0\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_6\}, \{q_5\}\}$

A condição de parada não foi satisfeita, e em seguida o algoritmo retorna para a linha 3 de forma a fazer uma nova iteração de refinamento. Nesta nova tentativa,  $P_{novo}$  não será mais refinado e consequentemente não seria modificado (deixa-se esta verificação a cargo do leitor). Assim, quando é novamente testada a equivalência entre  $P$  e  $P_{novo}$ , o resultado é verdadeiro e o algoritmo encerra (linha 14).

Desta forma, ao término da execução do algoritmo é obtido o particionamento  $P = \{\{q_0\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_6\}, \{q_5\}\}$ , indicando que  $q_1 \equiv q_2$  e  $q_3 \equiv q_4$ . Estas equivalências permitem remodelar o DFA da Figura 6.12(a) para aquele apresentado na Figura 6.12(b). Observe ainda que a eliminação do estado  $q_6$  não foi resultado da minimização, mas sim da percepção de que ele é um estado inatingível.

## 6.5 Conclusões e leituras recomendadas

Turing propôs um modelo abstrato de computação, conhecido como máquina de Turing, com o objetivo de explorar os limites da capacidade de expressar soluções de problemas. Sob esta ótica, a máquina de Turing é uma proposta de definição formal da noção intuitiva de algoritmo. Por conseguinte, trata-se de uma tentativa de responder ao questionamento sobre o que pode, e o que *não* pode ser computado.

Neste capítulo foram apresentados modelos matemáticos diferentes que executam tarefas neste contexto: decidir, semi-decidir, reconhecer, construir funções. A adição de certos aspectos (quantidade de fitas, leitoras e acesso aleatório) não aumenta o conjunto de tarefas que podem ser executadas por uma máquina simples de Turing. A capacidade de computação representada pela máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação. Qualquer outra forma de expressar algoritmos terá, no máximo, a mesma capacidade computacional da máquina de Turing.

## 6.6 Exercícios

**6.1** Construir um autômato finito determinístico que reconheça as linguagens a seguir:.

- (a)  $L(M) = \{w \in \{a, b\}^* \mid \text{cada } a \text{ em } w \text{ é imediatamente precedido por um } b\}$
- (b)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ tenha } abab \text{ como substring}\}$
- (c)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ tem dois } a\text{'s consecutivos ou dois } b\text{'s consecutivos}\}$
- (d)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ não tem } aa \text{ nem } bb \text{ como substrings}\}$
- (e)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ é uma string com número ímpar de } b\text{'s}\}$
- (f)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ contém um número ímpar de } a\text{'s e ímpar de } b\text{'s em qualquer ordem}\}$
- (g)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ tem } ab \text{ e } ba \text{ como substrings}\}$
- (h)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ contém um número ímpar de } a\text{'s e par de } b\text{'s em qualquer ordem}\}$
- (i)  $L(M) = \{w \in \{a, b\}^* \mid w \text{ inicia e termina com o mesmo símbolo}\}$

**6.2** Descreva a linguagem aceita por cada um dos autômatos da Figura 6.13.

**6.3** Construa um autômato finito determinístico cujo alfabeto são os dígitos decimais  $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  tal que os strings sejam divisíveis por três.

**6.4** Construa um autômato finito determinístico cujo alfabeto são os dígitos decimais  $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  tal que os strings sejam divisíveis por seis.

**6.5** Construa um autômato finito determinístico que reconheça se um número binário é divisível por três. A leitura do número começa pelo bit mais significativo (MSB) e termina no bit menos significativo (LSB).

**6.6** Construa um autômato finito determinístico que reconheça um número binário cujo divisão por cinco resulte em resto dois. A leitura do número começa pelo bit mais significativo (MSB) e termina no bit menos significativo (LSB).

**6.7** Construa um autômato finito determinístico que reconheça se um número binário é divisível por cinco. A leitura do número começa pelo bit mais significativo (MSB) e termina no bit menos significativo (LSB).

**6.8** Construa um autômato finito determinístico sobre o alfabeto  $\Sigma = \{0, 1\}$  que aceite strings que terminem com 01.

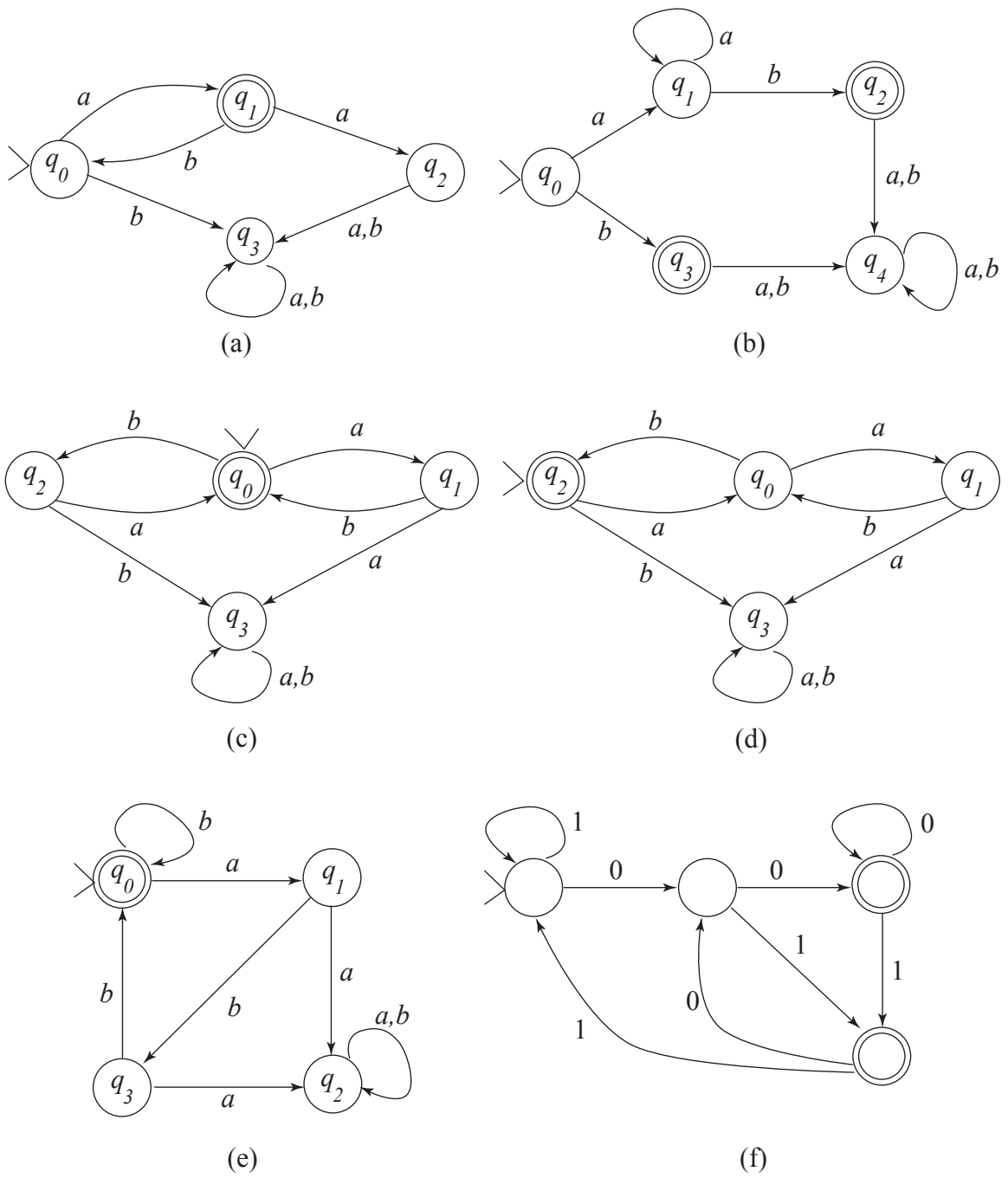


Figura 6.13: Autômatos do Exercício 6.2.



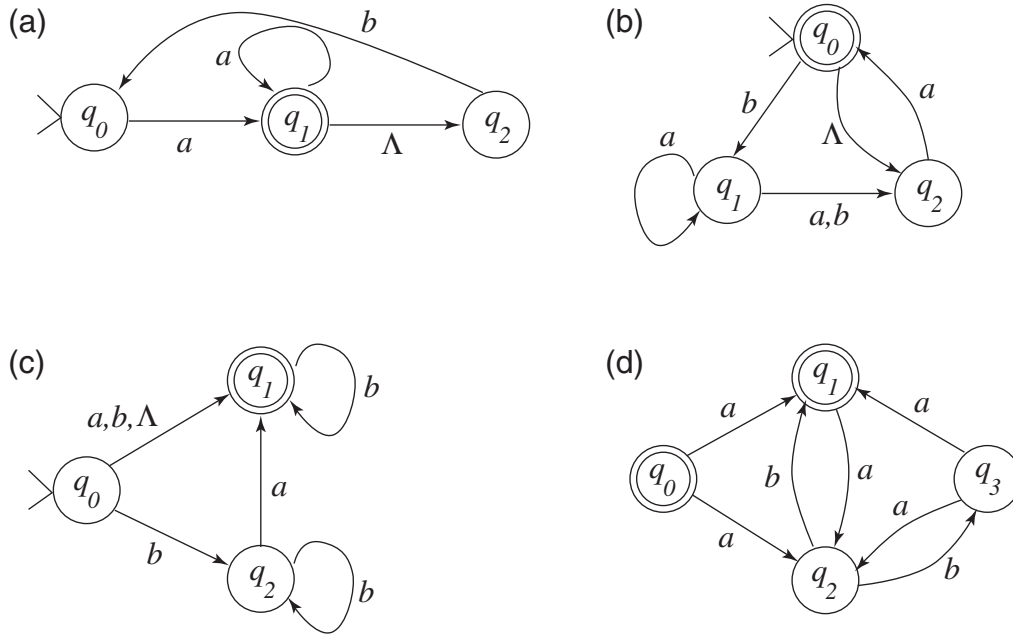


Figura 6.14: Autômatos do Exercício 6.13.

**6.9** Construa um autômato finito determinístico sobre o alfabeto  $\Sigma = \{0, 1\}$  que aceite strings que contenham uma quantidade igual de ocorrências de 01 e 10.

**6.10** Construa um autômato finito determinístico sobre o alfabeto  $\Sigma = \{0, 1\}$  que aceite strings que contenham pelo menos dois 0s e no máximo um 1.

**6.11** Construa um autômato finito determinístico sobre o alfabeto  $\Sigma = \{0, 1\}$  que aceite strings em que cada 0 do string seja imediatamente precedido e imediatamente seguido por 11.

**6.12** Determine o DFA correspondente ao NFA da Figura 6.6(a).

**6.13** Determine o DFA correspondente ao NFA da Figura 6.14.

**6.14** Determine o DFA correspondente ao NFA da Figura 6.15.

**6.15** Determine o DFA correspondente ao NFA da Figura 6.16.

**6.16** Realize a minimização dos autômatos da Figura 6.17.

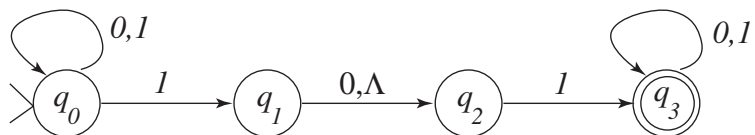


Figura 6.15: Autômato do Exercício 6.14.

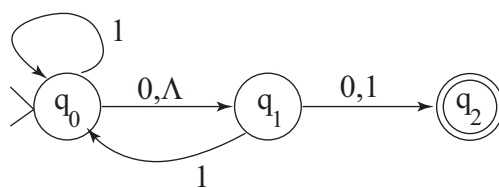


Figura 6.16: Autômato do Exercício 6.15.

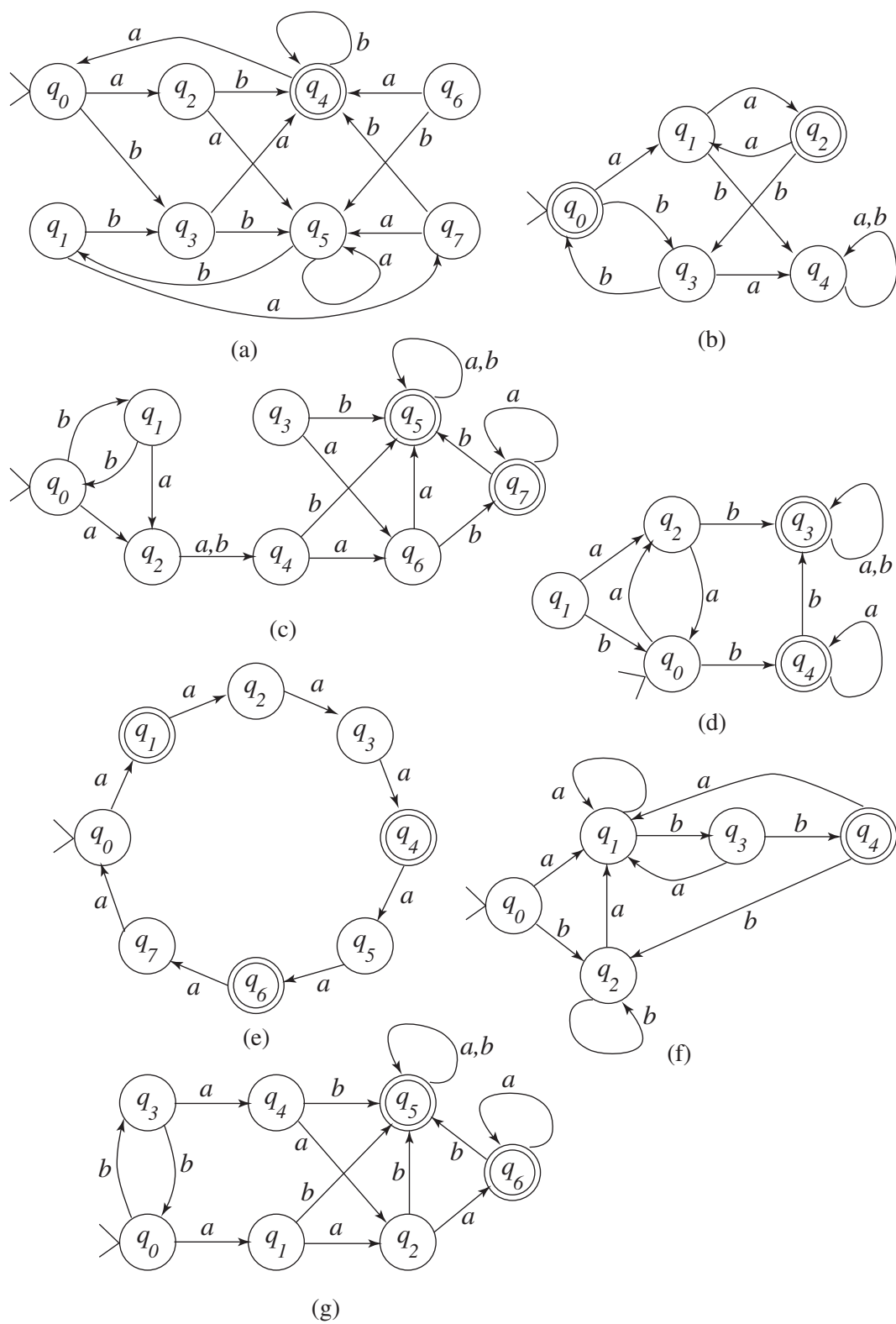


Figura 6.17: Autômato do Exercício 6.16.

# Referências Bibliográficas

- [1] HOPCROFT, J. E., *Theory os Machines and Compuatations*, chapter An n log n algorithm for minimizing states in a finite automaton, Academic Press, pp. 189–196, 1971.
- [2] MONTEIRO, A. A., PAULO, J. D. S., *Aritmética Racional*. Lisboa, Livraria Avelar Machado, 1945.
- [3] IFRAH, G., *Os Números: a história de uma grande invenção*. São Paulo, Globo S.A., 2005.
- [4] DEDEKIND, R., *Was sind und was sollen die Zahlen*, v. 3, *Gesammelte Mathematische Werke*. New York, Chelsea Publishing Company, 1969. pp. 335-391.
- [5] GÖDEL, K., *Collected Works*, v. 2, *Gesammelte Mathematische Werke*. Oxford, Oxford University Press, 1990.
- [6] ISRAEL, D., “Reflections on Gödel’s and Gandy’s Reflection on Turing’s Thesis”, *Minds and Machines*, v. 12, n. 2, pp. 181–201, 2002.
- [7] SOBRINHO, J. Z., “Aspectos da Tese de Church-Turing”, *Revista Matemática Universitária - USP*, v. 1, n. 6, pp. 1–23, 1987.
- [8] MCDERMOTT, D., “Artificial Intelligence Meets Natural Stupidity”, *SI-GART Newsletter*, v. 57, pp. 4–9, April 1976.
- [9] SETTI, M. D. O. G., *O Processo de Discretiza çã o do Raciocínio Matemático na Tradu çã o para o Raciocínio Computacional*, Report, Universidade Federal do Paraná, 2009.

- [10] GUERREIRO, G., “A Vanguarda Matemática e os Limites da Razão”, *Scientific American Brasil*, v. 5, n. 12, pp. 39–56, 2007. Coleção Gênios da Ciência.
- [11] SMULLYAN, R., *Gödel’s Incompleteness Theorems*. Oxford University Press, 1992.
- [12] GÖDEL, K., *The Undecidable*, chapter On Formally Undecidable Propositions of Principia Mathematica and Related Systems, New York, Raven Press, pp. 5–38, 1965.
- [13] WHITEHEAD, A. N., RUSSELL, B., *Principia Mathematica*. Londres, Cambridge University Press, 1913.
- [14] NAGEL, E., NEWMAN, J. R., *Gödel’s Proof*. USA, Routledge, 1989.
- [15] GÖDEL, K., “Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme - On Formally Undecidable Propositions of Principia Mathematica and Related Systems”, *Kurt Gödel: Collected Works*, v. 1, n. 1, pp. 144–195, 1986. Tradução para o inglês por Martin Hirzel. [www.research.ibm.com/people/h/hirzel/papers/canon00-goedel.pdf](http://www.research.ibm.com/people/h/hirzel/papers/canon00-goedel.pdf) [capturado em 13 de agosto de 2011].
- [16] MELO, A. C. V. D., SILVA, F. S. C. D., *Modelos Clássicos de Computação*, Coleção Schaum. São Paulo, Thomson, 2006.
- [17] KUBRUSLY, R. D. S., *Uma viagem informal ao Teorema de Gödel ou (O preço da matemática é o eterno matemático)*, Report, Universidade Federal do Rio de Janeiro, 2007. IM/UFRJ.
- [18] SMULLYAN, R., *Recursion Theory for Metamathematics*. Oxford University Press, 1993.
- [19] SMULLYAN, R., *What’s the Name of This Book*. Penguin Books, 1978.
- [20] SMULLYAN, R., *Forever Undecided: A Puzzle Guide to Gödel*. Oxford University Press, 1987.
- [21] GOLDSTEIN, R., *Incompleteness: The Proof and Paradox of Kurt Gödel*. W. W. Norton Company, Inc., 2005.

- [22] HOFSTADTER, D. R., *Gödel, Escher e Bach: an Eternal Golden Braid*. Nova Iorque, Basic Books, 1979.
- [23] Rodríguez-Consuegra, F. A. (ed.), *Kurt Gödel - Unpublished Philosophical Essays*. Berlin, Birkhäuser Verlag, 1995.
- [24] Feferman, S., *et al.* (eds.), *Kurt Gödel - Collected Works*, v. I, II, III. New York, Oxford University Press, 1986.
- [25] WANG, H., *Reflections on Kurt Gödel*. Cambridge, Massachusetts, The MIT Press, 1988.
- [26] SUPPES, P., *Axiomatic Set Theory*. New York, Dover, 1972.
- [27] LIPSCHUTZ, S., *Teoria Elementar dos Conjuntos*, Coleção Schaum. São Paulo, McGraw-Hill do Brasil, 1990.
- [28] SUPPES, P., *Axiomatic Theory Set*. USA, Van Nostrand Company Inc., 1960.
- [29] MELLO, F. L. D., CARVALHO, R. L. D., “Knowledge Geometry”, *Journal of Information and Knowledge Management*, v. 14, pp. 1550028, 2015.
- [30] STOLL, R. R., *Set Theory and Logic*. USA, Dover Publications Inc., 1961.
- [31] MIRAGLIA, F., *Teoria dos Conjuntos: um mínimo*. Brasil, Edusp - Editora da Universidade de São Paulo, 1992.
- [32] HALMOS, P., *Teoria Ingênua dos Conjuntos*. Brasil, Edusp - Editora da Universidade de São Paulo, 1970.
- [33] GRATZER, G., *Universal Algebra*. USA, Van Nostrand Company Inc., 1968.
- [34] COHN, P. M., *Universal Algebra*. USA, Harper and Row, 1965.
- [35] GALLIER, J. H., *Logic for Computer Science*. USA, John Wiley and Sons Inc., 1987.
- [36] PREPARATTA, F. P., YEH, R. T., *Introduction to Discrete Structures for Computer Science and Engineering*. USA, Addison-Wesley Publishing Company, 1973.

- [37] SHOENFIELD, J. R., *Degrees of Unsolvability*. North-Holland Publishing Company, 1971.
- [38] BRAINERD, W. S., LANDWEBER, L. H., *Theory of Computation*. USA, John Wiley and Sons, 1974.
- [39] EILENBERG, S., ELGOT, C., *Recursiveness*. New York: Academic Press, 1970.
- [40] CARVALHO, R. L. D., OLIVEIRA, C. M. G. M. D., *Modelos de Computação e Sistemas Formais*, 11<sup>a</sup> Escola de Computação. Rio de Janeiro, Universidade Federal do Rio de Janeiro, 1998.
- [41] BRAINERD, W. S., LANDWEBER, L. H., *Theory of Computation*. John Wiley & Sons, 1974.
- [42] KLEENE, S. C., *Introduction to Metamathematics*. D. Van Nostrand Company, Inc., 1952.
- [43] Rogers Jr., H., *Theory of Recursive Functions and Effective Computability*. USA, McGraw-Hill Book Company, 1967.
- [44] DAVIS, M., *Computability and Unsolvability*. New York, Dover, 1983.
- [45] BOOLOS, G. S., JEFFREY, R. C., *Computability and Logic*. Cambridge University Press, 1974.
- [46] MARTIN DAVIS, R. S., WEYUKER, E. J., *Computability Complexity and Languages*. New York, Academic Press, 1994.
- [47] MALLOZZI, J. S., LILLO, N. J. D., *Computability with Pascal*. New Jersey, Prentice-Hall, Inc., 1984.
- [48] TURING, A. M., *The Undecidable*, chapter On Computable Numbers, with an Application to the Entscheidungsproblem, New York, Raven Press, pp. 115–151, 1965.
- [49] ELGOT, C. C., ROBINSON, A., “Random-access stored-program machines, an approach to programming languages”, *Journal of the ACM*, v. 11, pp. 365–399, 1964.

- [50] PREPARATTA, F. P., YEH, R. T., *Introduction to Discrete Structures for Computer Science and Engineering*. Addison-Wesley Publishing Company, 1973.
- [51] HENNIE, F., *Introduction to Computability*. Addison-Wesley Publishing Company, 1977.
- [52] NELSON, R. J., *Introduction to Automata*. USA, Jonh Wiley & Sons, Inc., 1968.
- [53] CARVALHO, R. L. D., *Máquinas, Programas e Algoritmos*, 2<sup>a</sup> Escola de Computação. Campinas, Universidade Estadual de Campinas, 1981.
- [54] MENDELSON, E., *Introduction to Mathematical Logic*, Cole Mathematics Series. 3 ed. The Wadsworth and Brooks, 1987.
- [55] MANIN, Y. I., *A Course in Mathematical Logic*, Graduate Texts in Mathematics 53. 1 ed. Springer-Verlag, 1977.
- [56] HOMER, S., SELMAN, A. L., *Computability and Complexity Theory*, Texts in Computer Science. 2 ed. Springer, 2011.
- [57] CUTLAND, N. J., *Computability: An introduction to recursive function theory*. Cambridge University Press, 1980.
- [58] SIPSER, M., *Introduction to The Theory of Computation*, Course Technology Series. 2 ed. Thomson, 2006.
- [59] WALTER CARNIELLI, R. L. E., *Computability: computable functions, logic and the foundations of mathematics*. Belmont, Wadsworth and Brooks, 1989.
- [60] TARSKI, A., *Logic, semantics, metamathematics*, chapter Fundamental concepts of the methodology of the deductive sciences, London, Oxford at the Clarendon Press, pp. 60–109, 1969.
- [61] ADAM YOUNG, M. Y., *Malicious Cryptography: Exposing Cryptovirology*. John Wiley and Sons Inc., 2004.



- [62] BONFANTE, G., KACZMAREK, M., MARION, J.-Y., *A Classification of Viruses through Recursion Theorems*, volume 4497 of Lecture Notes in Computer Science. 2 ed. CiE 2007, 2007.
- [63] MACHTEY, M., YOUNG, P., *An Introduction to the General Theory of Algorithms*. New York, North Holland, 1978.
- [64] ROBINSON, J. A., “A machine oriented logic based on the resolution principle”, *J. Assoc. Comput.*, v. 12, pp. 23–41, 1965.
- [65] ROBINSON, J. A., “Automatic deduction with hyper-resolution”, *Internat. J. Comput. Math.*, v. 1, pp. 227–234, 1965.
- [66] GILMORE, P. C., “A proof method for quantification theory: Its justification and realization”, *IBM Journal of Research and Development*, v. 4, n. 1, pp. 28–35, 1960.
- [67] DAVIS, M., PUTNAM, H., “A computing procedure for quantification theory”, *Journal of the ACM*, v. 7, n. 3, pp. 201–215, 1960.
- [68] CHANG, C.-L., LEE, R. C.-T., *Symbolic Logic and Mechanical Theorem Proving*. Academic Press Inc, 1973.
- [69] KLEENE, S. C., *Mathematical Logic*. Wiley, 1967.
- [70] HILBERT, D., ACKERMANN, W., *Prinmciples of Mathematical Logic*. Chelsea, 1950.
- [71] MCCAWLEY, J. D., *Everything That Linguists Have Always Wanted To Know About Logic*. 2 ed. The University of Chicago Press, 1993.
- [72] SUPPES, P., *Introduction to Logic*. D. van Nostrand, 1966.
- [73] RUSSELL, B., *A Filosofia do Atomismo Lógico*, *Lógica e Conhecimento*. 1 ed. Abril Cultural, 1974. (Os Pensadores, 42).
- [74] RUSSELL, B., *Significado e Verdade*. 1 ed. Zahar, 1978.
- [75] POPPER, K. R., *A Lógica da Pesquisa Científica*. 2 ed. Cultrix, 1974.

- [76] LAKATOS, I., , MUSGRAVE, A., *A Crítica e o Desenvolvimento do Conhecimento*. 1 ed. EDUSP, Cultrix, 1979. Tradução: M. O. Caiado.
- [77] WANG, H., *From Mathematics to Philosophy*. 1 ed. Cultrix, 1974.
- [78] GREEN, C. C., *The Application of Theorem Proving to Question-Answering Systems*. Ph.D. dissertation, Stanford, June 1969. AI Project MEMO AI-96.
- [79] LOVELAND, D. W., *Automated Theorem Proving: a Logical Basis*. 1 ed. North Holland, 1978.
- [80] HUGHES, G. E., LONDEY, D. G., *The Elements of Formal Logic*. USA, Methuen and Co Ltd, 1965.
- [81] BOOK, R. V., OTTO, F., *String-Rewriting Systems*. USA, Springer-Verlag, 1993.
- [82] HOPCROFT, J. E., ULLMAN, J. D., *Introduction to Automata Theory, Language and Computation*. USA, Addison-Wesley Publishing Company, 1979.
- [83] HOPCROFT, J. E., ULLMAN, J. D., *Formal Languages and their Relation to Automata*. USA, Addison-Wesley Publishing Company, 1969.
- [84] CURRY, H. B., *Foundations of Mathematical Logic*. New York, Academic Press, 1977.
- [85] SMULLYAN, R., *Theory of Formal Systems*. USA, Princeton, 1961.
- [86] BERSTEL, J., BOASSON, L., CARTON, O., *et al.*, *Handbook of Automata: from Mathematics to Applications*, chapter Minimization of automata, European Mathematical Society, pp. 189–196, 2010.
- [87] BEAL, M. P., CROCHEMORE, M., “Minimizing incomplete automata”, *Workshop on Finite State Methods and Natural Language Processing*, , september 2008. Ispra.
- [88] VALMARI, A., LEHTINEN, P., “Efficient minimization of DFAs with partial transition”, *Proc. 25th Symp. Theoretical Aspects of Comp. Sci.*, v. 08001, pp. 645–656, 2008. S. Albers and P. Weil, editors.

- [89] PAPADONIKOLAKIS, M., BOUGANIS, C.-S., CONSTANTINIDES, G.,  
“Performance comparison of GPU and FPGA architectures for the SVM training problem”, *IEEE International Conference on FieldProgrammable Technology*, pp. 388–391, 2009.
- [90] MU, S., WANG, C., LIU, M., *et al.*, “Evaluating the potential of graphics processors for high performance embedded computing”, *Proc. IEEE Design, Automation and Test in Europe Conference and Exhibition*, pp. 709–714, 2011.
- [91] KAI HWANG, F. A. B., *Computer Architecture and Parallel Processing*. McGraw-Hill, 1984.
- [92] AGARWAL, P., KRISHNAN, S., MUSTAFA, N., *et al.*, *Algorithms - ESA 2003, Lecture Notes in Computer Science*, chapter Streaming Geometric Optimization Using Graphics Hardware, Springer Berlin-Heidelberg, pp. 115–151, 2003.
- [93] TANENBAUM, A. S., *Organização Estruturada de Computadores*. 3 ed. Prentice Hall do Brasil, 1997.
- [94] BACKUS, J., “Can Programming be Liberated from von Neumann Style? A functional style and its algebra of program”, *ACM Turing Award Lecture, Communications of the ACM*, v. 21, n. 8, pp. 613–641, 1978.
- [95] OWENS, J. D., LUEBKE, D., GOVINDARAJU, N., *et al.*, “A Survey of General-Purpose Computing on Graphics Hardware”, *Eurographics 2005, State of the Art Reports*, pp. 21–51, 2005.
- [96] GUSTAFSON, J. L., “Reevaluating Amdahl’s law”, *Communications of the ACM*, v. 5, n. 31, pp. 532, 1988.
- [97] HANDLER, W., *Parallel Processing Systems, an advanced course*, chapter Innovative computer architecture - how to increase parallelism but not complexity, Cambridge University Press, pp. 1–41, 1982.
- [98] LOBUR, J., NULL, L., *The Essentials of Computer Organization And Architecture*. Jones and Bartlett Pub, 2006.

- [99] LEWIS, H. R., PAPADIMITRIOU, C. H., *Elements of the Theory of Computation*. 2 ed. New York, Prentice-Hall, 1998.
- [100] DUNNE, P., *Computability Theory: Concepts and Applications*. Ellis Horwood, 1991.
- [101] AARONSON, S., “NP-complete Problems and Physical Reality”, *ACM SIGACT News*, , march 2005. Complexity Theory Column 46.