Departamento de Ciência da Computação

Instituto de Matemática e Estatística

Universidade de São Paulo

# Computação Quântica: Complexidade e Algoritmos

Carlos H. Cardonha

Marcel K. de Carli Silva

Cristina G. Fernandes
(orientadora)

Iniciação Científica: Agosto/2003 a Dezembro/2004

Apoio Financeiro da FAPESP 03/13236-0 e 03/13237-7

# Sumário

# Introdução

Em 1900, em uma palestra marcante no Congresso Internacional de Matemáticos realizado em Paris, Hilbert postulou 23 problemas matemáticos, que tratam de temas diversos em matemática e áreas afins. O décimo problema na lista de Hilbert (*determination of the solvability of a diophantine equation*) pergunta se é possível determinar se uma equação diofantina arbitrária tem ou não solução por meio de um "processo finito":

> *Given a diophantine equation with any number of unknown quantities and with rational numerical coefficients: to devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.*

Esse problema pode ser postulado em uma linguagem mais atual como o seguinte: existe um algoritmo que, dada uma equação diofantina, determina se esta tem ou não solução?

Note que a questão postulada por Hilbert precede de décadas a invenção de computadores. Foi apenas nos anos 30 que tais questões foram formuladas e tratadas dentro do que ficou depois conhecido como *teoria da computabilidade*. Esta é a parte da teoria da computação especializada em lidar com esse tipo de questão.

Foi nos anos 30, após um trabalho de Gödel em lógica, que a idéia de algoritmo começou a ser formalizada. Gödel [Göd31] introduziu o conceito de *função primitiva recursiva* como uma formalização dessa idéia. Church [Chu33, Chu36] introduziu o $\lambda$-*cálculo* e Kleene [Kle36] definiu o conceito de *funções recursivas parciais* e mostrou a equivalência entre esse e o $\lambda$-cálculo. Turing [Tur36, Tur37] por sua vez propôs a sua formalização da idéia de algoritmo: as chamadas *máquinas de Turing*. Nesses trabalhos, Turing mostrou também a equivalência do conceito de máquinas de Turing e de funções recursivas parciais. Vale mencionar que o conceito de máquinas de Turing foi independentemente proposto por Post [Pos36], um professor de colegial de Nova Iorque. Cada uma dessas propostas diferentes do conceito de algoritmo é chamada de *modelo de computação*.

Foi Kleene [Kle52] quem chamou de *tese de Church* a afirmação de que todo modelo de computação *razoável* é equivalente ao da máquina de Turing. A afirmação é propositalmente vaga, pois visa capturar mesmo modelos que ainda venham a ser propostos, e cuja natureza não podemos prever. Por *razoável* entende-se um modelo que seja realista, no sentido de poder (mesmo que de maneira aproximada) ser construído na prática.

A teoria da computabilidade no fundo diferencia os problemas *decidíveis* (para os quais existe um algoritmo) dos *indecidíveis* (para os quais não existe um algoritmo). O surgimento dos computadores nas décadas de 30 e 40 aos poucos evidenciou uma diferença entre os problemas decidíveis: muitos parecem ser bem mais difíceis que outros, no sentido de que se conhece apenas algoritmos extremamente lentos para eles. Com isso, surgiu a necessidade de refinar a teoria de computabilidade para tentar explicar essas diferenças. Foi apenas nos anos 60 que a *teoria de complexidade*, que trata de tais questões, tomou corpo, com a formalização da idéia de *algoritmo eficiente*, independentemente introduzida por Cobham [Cob65] e Edmonds [Edm65], e a proposta de reduções eficientes entre problemas, feita por Karp [Kar72].

Foi nessa época que surgiram as definições das classes de complexidade **P** e **NP** e do conceito de **NP**-*completude*, que captura de certa maneira a dificuldade de se conseguir algoritmos eficientes para certos problemas. Grosseiramente, um problema em **NP** é dito **NP**-*completo* se qualquer outro problema da classe **NP** pode ser reduzido eficientemente a ele. A mais famosa questão na área de teoria da computação é se **P** é ou não igual a **NP**. Se for mostrado que algum problema **NP**-completo está em **P**, então tal questão é resolvida e fica provado que $\mathbf{P} = \mathbf{NP}$.

Um marco na teoria de complexidade é o *teorema de Cook* [Coo71, Lev73], que prova a existência de problemas **NP**-completos. Cook mostrou que o problema conhecido como SAT, de decidir se uma fórmula booleana em forma normal conjuntiva é ou não satisfatível, é **NP**-completo. Após o teorema de Cook e o trabalho de Karp [Kar72], que mostrou que vários outros problemas conhecidos são **NP**-completos, essa teoria se desenvolveu amplamente, tendo estabelecido a dificuldade computacional de problemas das mais diversas áreas, como mostram Garey e Johnson [GJ79].

Um dos problemas mais famosos cuja complexidade continua em aberto, mesmo após várias décadas de esforço da comunidade no sentido de resolvê-lo, é o *problema da fatoração de inteiros*: dado um inteiro, determinar a sua fatoração em números primos. Recentemente, o seu parente próximo, o problema de decidir se um número inteiro é primo ou não, chamado de *problema da primalidade*, teve sua complexidade totalmente definida, com o algoritmo AKS, de Agrawal, Kayal e Saxena [AKS02a, AKS02b]. Esse algoritmo mostra que o problema da primalidade está na classe **P**, resolvendo com isso uma questão em aberto há anos. Não se sabe até hoje, no entanto, se há um algoritmo eficiente para resolver o problema da fatoração de inteiros!

Na verdade, a dificuldade computacional do problema da fatoração de inteiros tem sido usada de maneira crucial em alguns sistemas criptográficos bem-conhecidos. Se for descoberto um algoritmo eficiente para resolver o problema da fatoração, vários sistemas criptográficos importantes seriam quebrados, incluindo o famoso sistema RSA de chave pública, criado por Rivest, Shamir e Adleman [RSA78].

O assunto de nossa iniciação científica — Computação Quântica — trata de um novo modelo de computação, o modelo quântico, que vem levantando questões intrigantes dentro da teoria de complexidade, e pode ter impactos práticos dramáticos no mínimo na área de criptologia. O modelo quântico de computação não infringe a validade da tese de Church, porém questiona a validade de uma versão mais moderna dessa, a chamada *tese de Church estendida*, que diz que todo modelo de computação razoável pode ser simulado *eficientemente* por uma máquina de Turing.

Pode-se dizer que a teoria de computação quântica iniciou-se nos anos 80, quando Feynman [Fey82] observou que um sistema quântico de partículas, ao contrário de um sistema clássico, parece não poder ser simulado eficientemente em um computador clássico, e sugeriu um computador que explorasse efeitos da física quântica para contornar o problema. Desde então, até 1994, a teoria de computação quântica desenvolveu-se discretamente, com várias contribuições de Deutsch [Deu85, Deu89], Bernstein e Vazirani [BV97], entre outros, que colaboraram fundamentalmente para a formalização de um modelo computacional quântico.

Foi apenas em 1994 que a teoria recebeu um forte impulso e uma enorme divulgação. Isso deveu-se principalmente ao algoritmo de Shor [Sho94, Sho97], um algoritmo quântico eficiente para o problema da fatoração de inteiros, considerado o primeiro algoritmo quântico combinando relevância prática e eficiência. O algoritmo de Shor é uma evidência de que o modelo computacional quântico proposto pode superar de fato o modelo clássico, derivado das máquinas de Turing. O resultado de Shor impulsionou tanto a pesquisa prática, objetivando a construção de um computador segundo o modelo quântico, quanto a busca por algoritmos criptográficos alternativos e algoritmos quânticos eficientes para outros problemas difíceis. Essas e várias outras questões, relacionadas tanto com a viabilidade do modelo quântico quanto com as suas limitações, têm sido objeto de intensa pesquisa científica.

Do ponto de vista prático, busca-se descobrir se é ou não viável construir um computador segundo o modelo quântico que seja capaz de manipular números suficientemente grandes. Tal viabilidade esbarra em uma série de questões técnicas e barreiras físicas e tecnológicas. Já se tem notícia de computadores construídos segundo o modelo quântico, mas todos ainda de pequeno porte. Em 2001, por exemplo, foi construído um computador quântico com 7 *qubits* (o correspondente aos bits dos computadores tradicionais). Nesse computador, foi implementado o algoritmo de Shor que, nele, fatorou o número 15. Uma parte dos cientistas da computação acredita que a construção de computadores quânticos de maior porte será possível, enquanto outra parte não acredita nisso.

Do ponto de vista de teoria de complexidade, busca-se estabelecer a relação entre as classes de complexidade derivadas do modelo quântico e as classes de complexidade tradicionais. Também busca-se, claro, estabelecer a complexidade no modelo quântico de problemas bem conhecidos, ou seja, busca-se por algoritmos quânticos eficientes para outros problemas relevantes.

## Organização do texto

Este texto está dividido em três partes. A primeira concentra-se nos resultados algorítmicos da área, a segunda, nos tópicos de teoria de complexidade computacional e a terceira consiste em uma coleção de apêndices, cada um sobre um assunto que é um pré-requisito ou é relacionado aos tratados nas duas partes principais.

A parte algorítmica começa descrevendo os componentes básicos de um computador quântico e a formalização de suas operações básicas. Alguns algoritmos quânticos simples são apresentados e finalmente o algoritmo de Shor é descrito e analisado.

A formalização de um computador quântico utiliza fortemente espaços de Hilbert e algumas noções de mecânica quântica. O apêndice A trata de espaços de Hilbert e o apêndice B, de mecânica quântica. Esses apêndices podem ser ignorados, lidos ou consultados esporadicamente dependendo da bagagem prévia do leitor. Para a compreensão do algoritmo de Shor, vários resultados básicos de teoria dos números são necessários. O apêndice C apresenta todos os resultados relevantes de teoria dos números para o estudo do algoritmo de Shor e testes de primalidade ou algoritmos de fatoração em geral. Além disso, esse apêndice contém também a descrição e análise de várias rotinas básicas que são usadas na descrição de testes de primalidade e, em particular, no algoritmo de Shor. O apêndice D está também relacionado ao algoritmo de Shor, porém não como um pré-requisito, mas como uma curiosidade ou paralelo. Para leitores com menos familiaridade com ciência da computação, recomendamos a sua leitura. Nele, são apresentados e discutidos alguns resultados tradicionais de teste de primalidade. Uma seção curta deste apêndice comenta também o sistema RSA e sua relação com testes de primalidade e algoritmos de fatoração.

A segunda parte do texto trata de resultados de complexidade computacional. Ela começa apresentando as máquinas de Turing tradicionais e a máquina de Turing quântica. Um resultado bastante complexo porém importante é apresentado a seguir: a descrição de uma máquina de Turing quântica universal. Depois disso, são introduzidas as várias classes de complexidade envolvidas e são apresentados vários resultados de complexidade entre essas classes. Esses resultados mostram algumas das técnicas usadas na simulação de máquinas de Turing quânticas, e ressalta as dificuldades nessas simulações frente às simulações típicas do modelo clássico.

Esperamos com esse texto dar uma visão geral desta nova e intrigante área, tanto do ponto de vista algorítmico quanto do ponto de vista de teoria de complexidade.

# Parte I

# Aspectos Algorítmicos

# Capítulo 1

# Bits, registradores e circuitos quânticos

A apresentação dos resultados algorítmicos dessa área depende da formalização de alguns conceitos básicos. Introduziremos os conceitos de bits e registradores quânticos, que são os blocos fundamentais de um computador quântico, bem como a noção de circuitos quânticos, que correspondem ao conceito de algoritmo no modelo tradicional.

## 1.1 Bits quânticos

Seja $\mathcal{H}_2$ um espaço de Hilbert de dimensão 2. Fixe uma base ortonormal $B_2 := \big\{|0\rangle, |1\rangle\big\}$ de $\mathcal{H}_2$. Um *qubit* ou *bit quântico* é um vetor unitário em $\mathcal{H}_2$, isto é, um vetor $|\phi\rangle \in \mathcal{H}_2$ é um qubit se

$$|\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \tag{1.1}$$

com $\alpha_0, \alpha_1 \in \mathbb{C}$ e $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Dizemos que os vetores $|0\rangle$ e $|1\rangle$ são os *estados básicos* e que o qubit $|\phi\rangle$ está numa *superposição* de estados básicos (contraste isto com o modelo clássico, onde um bit assume apenas um dos valores 0 ou 1). Chamamos ao coeficiente complexo $\alpha_j$ de *amplitude* do estado básico $|j\rangle$, para $j = 0, 1$.

No modelo clássico, se tivermos em mãos um bit $b$, podemos descobrir sem problemas se $b$ vale 0 ou 1 e isso em nada afeta o valor de $b$. Já no modelo quântico, não é possível determinar o valor de um qubit $|\phi\rangle$. Se tentarmos, o que observamos é o resultado de um evento probabilístico que tem como efeito colateral a alteração irreversível do valor de $|\phi\rangle$. Mais precisamente, ao medirmos o estado de um qubit $|\phi\rangle$ dado pela equação (1.1), enxergaremos o "valor" $|0\rangle$ com probabilidade $|\alpha_0|^2$ e o "valor" $|1\rangle$ com probabilidade $|\alpha_1|^2$. Se o "valor" observado for $|0\rangle$, o estado do qubit $|\phi\rangle$, imediatamente após a medição, será $|0\rangle$, e analogamente se o estado observado for $|1\rangle$. Note então que, apesar de um qubit armazenar uma superposição de estados, usando medições, só conseguimos obter dele um dos estados da superposição.

Existem apenas duas portas lógicas operando sobre um bit clássico: a porta identidade e a negação. No modelo quântico de computação, qualquer transformação unitária em $\mathcal{H}_2$ é uma porta quântica. Uma matriz $U \in \mathbb{C}^{n \times n}$ é dita *unitária* se $U^*U = UU^* = I$, onde $I$ é a matriz identidade e $U^*$ é a transposta conjugada de $U$. Para trabalharmos com tais transformações, convencionamos que um qubit $|\phi\rangle$ dado pela equação (1.1) tem a seguinte representação na base $B_2$:

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = |\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle. \tag{1.2}$$

Assim, a matriz de Hadamard

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.3}$$

transforma o qubit $|0\rangle$ no qubit

$$H|0\rangle = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big). \tag{1.4}$$

Como no caso de circuitos tradicionais, será útil convencionarmos uma representação gráfica para circuitos quânticos, indicando a ordem de aplicação de portas e medições. Por exemplo, o circuito ilustrado na figura 1.1 indica que a matriz de Hadamard $H$ deve ser aplicada ao qubit $|\phi\rangle$ e depois o qubit resultante deve ser medido para obtermos o qubit $|\phi'\rangle$. Se $|\phi\rangle$ for inicializado com $|0\rangle$, então $|\phi'\rangle$ será $|0\rangle$ ou $|1\rangle$ equiprovavelmente, de acordo com a equação (1.4).

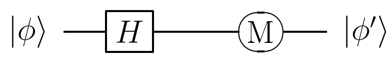


Figura 1.1: Um circuito quântico.

É interessante observar como as medições afetam o comportamento do circuito. Por exemplo, se inicializarmos $|\phi\rangle$ com $|0\rangle$, então $|\phi'\rangle$ no circuito quântico da figura 1.2 sempre será $|0\rangle$. Já no circuito da figura 1.3, o estado $|\phi'\rangle$ será $|0\rangle$ ou $|1\rangle$ equiprovavelmente.

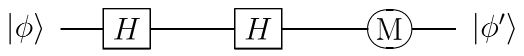


Figura 1.2: Mais um circuito.

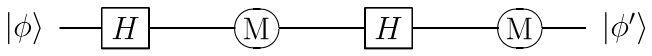


Figura 1.3: Circuito com medição intercalada.

### 1.1.1 ⋆ Representação gráfica de um qubit

Vamos apresentar uma possível representação gráfica para um qubit, que nos ajudará mais tarde a visualizar algumas operações sobre estes.

Esta subseção é opcional e só utilizamos os resultados aqui encontrados na subseção 1.1.2, também opcional.

**Proposição 1.1.** *Sejam $\theta \in \mathbb{R}$ e $A$ uma matriz tal que $A^2 = I$. Então*

$$e^{i\theta} = \cos\theta + i \operatorname{sen}\theta$$

*e*

$$e^{i\theta A} = \cos(\theta) I + i \operatorname{sen}(\theta) A.$$

*Demonstração.* Primeiro notamos que as expansões em séries de Taylor das funções $e^x$, $\operatorname{sen} x$ e $\cos x$ são

$$\begin{aligned}
e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^k}{k!} + \cdots = \sum_{k=0}^{\infty} \frac{x^k}{k!} \\
\operatorname{sen} x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^k \frac{x^{2k+1}}{(2k+1)!} + \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!} \\
\cos x &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + (-1)^k \frac{x^{2k}}{(2k)!} + \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}.
\end{aligned}$$

Mas então temos

$$\begin{aligned}
e^{i\theta} &= 1 + i\theta + \frac{(i\theta)^2}{2!} + \frac{(i\theta)^3}{3!} + \frac{(i\theta)^4}{4!} + \frac{(i\theta)^5}{5!} + \cdots \\
&= 1 + i\theta - \frac{\theta^2}{2!} - i\frac{\theta^3}{3!} + \frac{\theta^4}{4!} + i\frac{\theta^5}{5!} + \cdots \\
&= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} + \cdots\right) + i\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \cdots\right) \\
&= \cos\theta + i \operatorname{sen}\theta.
\end{aligned}$$

Similarmente, se expandirmos $e^A$ como

$$e^A := I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots + \frac{A^k}{k!} + \cdots,$$

teremos

$$\begin{aligned}
e^{i\theta A} &= I + i\theta A + \frac{(i\theta A)^2}{2!} + \frac{(i\theta A)^3}{3!} + \frac{(i\theta A)^4}{4!} + \frac{(i\theta A)^5}{5!} + \cdots \\
&= I + i\theta A - \frac{\theta^2}{2!}I - i\frac{\theta^3}{3!}A + \frac{\theta^4}{4!}I + i\frac{\theta^5}{5!}A + \cdots \\
&= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} + \cdots\right) I + i\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \cdots\right) A \\
&= \cos(\theta)I + i\,\mathrm{sen}(\theta)A.
\end{aligned}$$

□

Note então que, dado $z \in \mathbb{C}$, com $|z| = 1$, existe $\theta \in \mathbb{R}$ tal que $z = e^{i\theta}$. Para encontrarmos um $\theta$ satisfazendo essa igualdade, sejam $a$ e $b$ tais que $z = a + bi$. Note que os sinais de $a$ e $b$ determinam o quadrante em que $\theta$ se encontra. Em particular, se $a = 0$, é óbvio que $\theta$ só pode ser $\pi/2$ ou $-\pi/2$, de acordo com o sinal de $b$. Se $a \neq 0$, então $\theta$ é dado pelo ângulo $\mathrm{arctg}\, b/a$ no respectivo quadrante.

**Proposição 1.2.** *Seja $|\psi\rangle$ um qubit. Então existem $\gamma, \phi, \theta \in \mathbb{R}$ tais que $|\psi\rangle = e^{i\gamma}\Big(\cos\frac{\theta}{2}\ |0\rangle + e^{i\phi}\,\mathrm{sen}\,\frac{\theta}{2}\ |1\rangle\Big)$.*

*Demonstração.* Seja $|\psi\rangle := \alpha|0\rangle + \beta|1\rangle$ um qubit. Então temos que $\alpha, \beta \in \mathbb{C}$ e $|\alpha|^2 + |\beta|^2 = 1$. Segue que $|\alpha|^2 \leq 1$ e portanto $0 \leq |\alpha| \leq 1$.

Se $|\alpha| = 0$, então temos $\alpha = 0$ e $|\beta| = 1$. Neste caso, existe $\phi \in \mathbb{R}$ tal que $\beta = e^{i\phi}$. Tome $\gamma := 0$ e $\theta := \pi$ e estamos feitos.

Se $|\alpha| = 1$, então temos $\beta = 0$ e existe $\gamma \in \mathbb{R}$ tal que $\alpha = e^{i\gamma}$. Tome $\phi := 0$ e $\theta := 0$ e estamos feitos.

Senão, temos $0 < |\alpha| < 1$ e $0 < |\beta| < 1$. Tome $\theta := 2 \arccos |\alpha|$, de modo que $\cos(\theta/2) = |\alpha|$. Como $|\alpha|^2 + |\beta|^2 = 1$, segue que $|\beta|^2 = \mathrm{sen}^2(\theta/2)$ e portanto $|\beta| = \big|\mathrm{sen}(\theta/2)\big|$.

Tome $\alpha' := \alpha/\cos(\theta/2)$ e $\beta' := \beta/\mathrm{sen}(\theta/2)$. Pelo que foi visto acima, temos $|\alpha'| = |\beta'| = 1$. Então existem $\gamma, \delta \in \mathbb{R}$ tais que $\alpha' = e^{i\gamma}$ e $\beta' = e^{i\delta}$. Mas então

$$\begin{aligned}
|\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \;=\; \alpha' \cos(\theta/2)|0\rangle + \beta' \,\mathrm{sen}(\theta/2)|1\rangle \\
&= e^{i\gamma} \cos(\theta/2)|0\rangle + e^{i\delta}\,\mathrm{sen}(\theta/2)|1\rangle \\
&= e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i(\delta-\gamma)}\,\mathrm{sen}\,\frac{\theta}{2}|1\rangle\right).
\end{aligned}$$

Tome $\phi := \delta - \gamma$ e estamos feitos. $\square$

De acordo com os axiomas da mecânica quântica, apresentados na seção B.1, um bit quântico é simplesmente um estado quântico num sistema quântico representado por um espaço de Hilbert bidimensional. Vemos então, na proposição acima, que $e^{i\gamma}$ é um fator de fase global. Podemos então reenunciar este resultado de acordo com nossa discussão sobre estados quânticos indistingüíveis, feita na subseção B.1.1:

**Proposição 1.3.** *Seja $|\psi\rangle$ um qubit. Então existem $\varphi, \theta \in \mathbb{R}$ tais que $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\operatorname{sen}\frac{\theta}{2}|1\rangle$.*

Diante desta maneira de escrever um qubit $|\psi\rangle$, podemos representar graficamente $|\psi\rangle$ na esfera de Bloch (também chamada esfera de Poincaré), uma esfera de raio unitário. Como pode ser visto na figura 1.4, se $\vec{r}$ é a representação de $|\psi\rangle$, então a projeção de $\vec{r}$ sobre o plano $xy$ forma um ângulo de $\varphi$ com o eixo $x$ e $\theta$ é o ângulo de $\vec{r}$ com o eixo $z$.

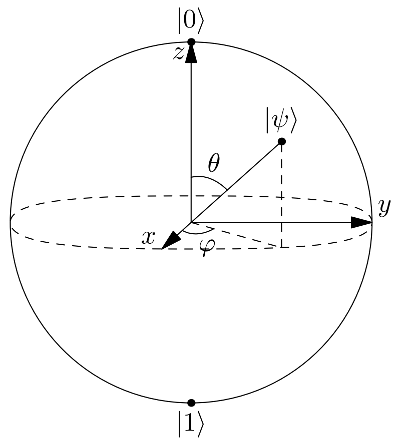


Figura 1.4: Representação de $|\psi\rangle := \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\operatorname{sen}\frac{\theta}{2}|1\rangle$ na esfera de Bloch.

### 1.1.2 $\star$ Decomposição de matrizes unitárias

Desenvolvemos nesta seção um resultado sobre decomposições de matrizes unitárias em $\mathbb{C}^{2\times 2}$ que mostra os "blocos fundamentais" de construção destas matrizes.

Esta subseção é opcional e depende dos resultados encontrados na subseção 1.1.1, também opcional.

Considere as seguintes matrizes, denominadas matrizes de Pauli:

$$\sigma_x := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.5}$$

Vamos mostrar que os "blocos fundamentais", que definimos a seguir, de construção de matrizes unitárias em $\mathbb{C}^{2\times 2}$ são gerados pelas matrizes de Pauli.

Seja $\theta \in \mathbb{R}$. Definimos, para $w = x, y, z$, a *matriz de rotação de $\theta$ sobre o eixo $w$* como

$$R_w(\theta) := e^{i\frac{\theta}{2}\sigma_w},$$

onde as matrizes $\sigma_x$, $\sigma_y$ e $\sigma_z$ foram definidas em (1.5) (observe que $\sigma_x^2 = \sigma_y^2 = \sigma_z^2 = I$).

Como indicado pelo nome, essas matrizes representam, na esfera de Bloch, uma rotação no sentido horário de um certo ângulo $\theta$ sobre um determinado eixo.

Note que tais matrizes de fato são unitárias, já que

$$\begin{aligned} [R_w(\theta)]^* R_w(\theta) &= \left(\cos\frac{\theta}{2}I - i\,\mathrm{sen}\,\frac{\theta}{2}\sigma_w^*\right)\left(\cos\frac{\theta}{2}I + i\,\mathrm{sen}\,\frac{\theta}{2}\sigma_w\right) \\ &= \cos^2\frac{\theta}{2}I + i\cos\frac{\theta}{2}\,\mathrm{sen}\,\frac{\theta}{2}(\sigma_w - \sigma_w^*) + \mathrm{sen}^2\,\frac{\theta}{2}I = I, \end{aligned}$$

pois $\sigma_w^* = \sigma_w$ para $w = x, y, z$.

Expandindo $R_w(\theta)$ para os três eixos, obtemos:

$$R_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & i\,\mathrm{sen}\,\frac{\theta}{2} \\ i\,\mathrm{sen}\,\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \qquad R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & \mathrm{sen}\,\frac{\theta}{2} \\ -\mathrm{sen}\,\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} e^{i\frac{\theta}{2}} & 0 \\ 0 & e^{-i\frac{\theta}{2}} \end{pmatrix}.$$

Essas transformações são importantes na decomposição de matrizes unitárias de dimensão 2 e recebem nomes especiais. Seguindo a terminologia estabelecida em Barenco *et al.* [BBC$^+$95], definimos as seguintes transformações sobre $\mathcal{H}_2$:

- $\mathrm{Rot}(\theta) := R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & \mathrm{sen}\,\frac{\theta}{2} \\ -\mathrm{sen}\,\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$, uma *rotação* de $\theta$;
- $\mathrm{Ph}(\alpha) := R_z(\alpha) = \begin{pmatrix} e^{i\frac{\alpha}{2}} & 0 \\ 0 & e^{-i\frac{\alpha}{2}} \end{pmatrix}$, uma *mudança de fase* de $\alpha$;
- $\mathrm{Scal}(\delta) := e^{i\delta}I = \begin{pmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{pmatrix}$, uma *"escala"* de $\delta$.

Além disso, a matriz $\sigma_x$ também é chamada de *"negação"*.

Com isso, estamos chamando de rotação apenas a rotação sobre o eixo $y$. A rotação sobre $z$ muda tão somente a fase de um vetor e por isso recebe o nome de mudança de fase (*phase shift*). Já a operação de "escala" apenas altera o fator de fase global.

Valem as seguintes propriedades, facilmente verificadas:

1. $\operatorname{Rot}(\theta_1)\operatorname{Rot}(\theta_2) = \operatorname{Rot}(\theta_1 + \theta_2)$;

2. $\operatorname{Ph}(\alpha_1)\operatorname{Ph}(\alpha_2) = \operatorname{Ph}(\alpha_1 + \alpha_2)$;

3. $\operatorname{Scal}(\delta_1)\operatorname{Scal}(\delta_2) = \operatorname{Scal}(\delta_1 + \delta_2)$;

4. $\sigma_x \operatorname{Rot}(\theta)\, \sigma_x = \operatorname{Rot}(-\theta)$;

5. $\sigma_x \operatorname{Ph}(\alpha)\, \sigma_x = \operatorname{Ph}(-\alpha)$.

Com isso, podemos provar o seguinte:

**Teorema 1.4.** *Seja $U \in \mathbb{C}^{2\times 2}$ uma matriz unitária. Então existem $\alpha, \beta, \gamma, \theta \in \mathbb{R}$ tais que*

$$U = e^{i\gamma} \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{-i\alpha} \end{pmatrix} \begin{pmatrix} \cos\theta & i \operatorname{sen}\theta \\ i \operatorname{sen}\theta & \cos\theta \end{pmatrix} \begin{pmatrix} e^{i\beta} & 0 \\ 0 & e^{-i\beta} \end{pmatrix}. \tag{1.6}$$

*Demonstração.* Vamos apresentar uma prova construtiva da existência da fatoração (1.6).

Seja

$$U = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix}$$

uma matriz unitária. Então suas colunas, que denotamos por $U_1$ e $U_2$, bem como suas linhas, são ortonormais. Logo temos que $\langle U_1 | U_1 \rangle = u_{11}^* u_{11} + u_{21}^* u_{21} = \|U_1\|^2 = 1$. Portanto

$$|u_{11}|^2 + |u_{21}|^2 = 1. \tag{1.7}$$

Similarmente, a primeira linha e a segunda coluna têm norma 1:

$$\begin{aligned} |u_{11}|^2 + |u_{12}|^2 &= 1 && (1.8) \\ |u_{12}|^2 + |u_{22}|^2 &= 1 && (1.9) \end{aligned}$$

e as colunas $U_1$ e $U_2$ são ortogonais, de forma que $\langle U_1 | U_2 \rangle = 0$, ou seja,

$$u_{11}^* u_{12} + u_{21}^* u_{22} = 0. \tag{1.10}$$

Por (1.7), temos que $|u_{11}|^2 \leq 1$. Segue que $0 \leq |u_{11}| \leq 1$.

Se $|u_{11}| = 0$, então $u_{11} = 0$. Além disso, por (1.7), $|u_{21}| = 1$; por (1.8), $|u_{12}| = 1$; finalmente, por (1.9), $|u_{22}| = 0$ e portanto $u_{22} = 0$. Então existem $a, b \in \mathbb{R}$ tais que $u_{21} = e^{ia}$ e $u_{12} = e^{ib}$. Tome $\alpha := 0$, $\beta := (a-b)/2$, $\gamma := (a+b-\pi)/2$ e $\theta := \pi/2$ e estamos feitos.

Se $|u_{11}| = 1$, então por (1.7) temos $|u_{21}| = 0$ e imediatamente $u_{21} = 0$. Além disso, por (1.8), $|u_{12}| = 0$ de forma que $u_{12} = 0$; por (1.9), $|u_{22}| = 1$. Então existem $a, b \in \mathbb{R}$ tais que $u_{11} = e^{ia}$ e $u_{22} = e^{ib}$. Tome $\alpha := 0$, $\beta := (a-b)/2$, $\gamma := (a+b)/2$ e $\theta := 0$ e estamos feitos.

Senão, $0 < |u_{11}| < 1$. Tome $\theta := \arccos |u_{11}|$, de forma que $\cos\theta = |u_{11}|$ e $0 < \theta < \pi/2$. Por (1.7), $|u_{21}|^2 = 1 - \cos^2\theta = \operatorname{sen}^2\theta$. Logo, $|u_{21}| = |\operatorname{sen}\theta|$. Semelhantemente, por (1.8), $|u_{12}| = |\operatorname{sen}\theta|$ e, por (1.9), $|u_{22}| = |\cos\theta|$. Note também que, como $0 < \theta < \pi/2$, então $\cos\theta \neq 0 \neq \operatorname{sen}\theta$.

Tome $a_{11} := u_{11}/(\cos\theta)$, $a_{21} := u_{21}/(i\operatorname{sen}\theta)$, $a_{12} := u_{12}/(i\operatorname{sen}\theta)$ e $a_{22} := u_{22}/(\cos\theta)$. Note que $|a_{11}| = \big(1/|\cos\theta|\big)\,|u_{11}| = 1$, $|a_{21}| = \big(1/|i\operatorname{sen}\theta|\big)\,|u_{21}| = 1$, $|a_{12}| = \big(1/|i\operatorname{sen}\ \theta|\big)\,|u_{12}| = 1$ e $|a_{22}| = \big(1/|\cos\theta|\big)\,|u_{22}| = 1$. Então existem $A_{11}, A_{12}, A_{21}, A_{22} \in \mathbb{R}$ tais que $a_{ij} = e^{iA_{ij}}$ para $1 \leq i, j \leq 2$.

Agora tome $\alpha := -(A_{21} - A_{11})/2$, $\beta := -(A_{12} - A_{11})/2$ e $\gamma := (A_{12} + A_{21})/2$. Vamos verificar que essa fatoração de $U$ é válida. Multiplicando todos os fatores de (1.6), obtemos

$$\begin{pmatrix} e^{i(\gamma+\alpha+\beta)}\cos\theta & e^{i(\gamma+\alpha-\beta)} i\operatorname{sen}\theta \\ e^{i(\gamma-\alpha+\beta)} i\operatorname{sen}\theta & e^{i(\gamma-\alpha-\beta)}\cos\theta \end{pmatrix},$$

de modo que precisamos verificar que $a_{11} = e^{i(\gamma+\alpha+\beta)}$, $a_{12} = e^{i(\gamma+\alpha-\beta)}$, $a_{21} = e^{i(\gamma-\alpha+\beta)}$ e $a_{22} = e^{i(\gamma-\alpha-\beta)}$.

Veja que

$$\gamma + \alpha + \beta = \frac{A_{12} + A_{21}}{2} - \frac{A_{21} - A_{11}}{2} - \frac{A_{12} - A_{11}}{2} = A_{11}$$

e então $a_{11} = e^{iA_{11}} = e^{i(\gamma+\alpha+\beta)}$. É semelhantemente fácil verificar que $\alpha, \beta$ e $\gamma$ também satisfazem as igualdades para $a_{12}$ e $a_{21}$. Para $a_{22}$, primeiro note que

$$\gamma - \alpha - \beta = \frac{A_{12} + A_{21}}{2} + \frac{A_{21} - A_{11}}{2} + \frac{A_{12} - A_{11}}{2} = A_{12} + A_{21} - A_{11}. \quad (1.11)$$

Agora vamos substituir em (1.10) os valores que temos para $U$, isto é, $u_{11}^* = e^{-iA_{11}}\cos\theta$, $u_{12} = e^{iA_{12}} i\operatorname{sen}\theta$, $u_{21}^* = (e^{iA_{21}} i\operatorname{sen}\theta)^* = (e^{i(A_{21}+\pi/2)}\operatorname{sen}\theta)^* =$

$e^{-i(A_{21}+\pi/2)} \operatorname{sen}\theta = -e^{-iA_{21}} i \operatorname{sen}\theta$ e $u_{22} = e^{iA_{22}} \cos\theta$:

$$\begin{aligned} & (e^{-iA_{11}} \cos\theta)(e^{iA_{12}} i \operatorname{sen}\theta) + (-e^{-iA_{21}} i \operatorname{sen}\theta)(e^{iA_{22}} \cos\theta) = 0 \\ \Rightarrow\ & i\cos\theta \cdot \operatorname{sen}\theta (e^{i(A_{12}-A_{11})} - e^{i(A_{22}-A_{21})}) = 0 \\ \Rightarrow\ & e^{i(A_{12}-A_{11})} = e^{i(A_{22}-A_{21})}, \text{ pois } i\cos\theta \cdot \operatorname{sen}\theta \neq 0 \\ \Rightarrow\ & e^{i(A_{12}-A_{11})} e^{iA_{21}} = e^{iA_{22}} \\ \Rightarrow\ & e^{i(A_{12}+A_{21}-A_{11})} = e^{iA_{22}}. \end{aligned}$$

Mas então, de (1.11), $e^{i(\gamma-\alpha-\beta)} = e^{i(A_{12}+A_{21}-A_{11})} = e^{iA_{22}} = a_{22}$ e estamos feitos.

□

Ou seja, toda matriz unitária de dimensão 2 pode ser escrita como uma composição de rotações sobre determinados eixos, mais uma mudança no fator de fase global. Mais especificamente, como $e^{i\gamma} R_z(\alpha) R_x(\theta) R_z(\beta)$. Seguindo os mesmos passos desta demonstração, podemos provar outra forma geral dessas matrizes:

**Teorema 1.5.** *Seja $U \in \mathbb{C}^{2\times 2}$ uma matriz unitária. Então existem $\alpha, \beta, \gamma, \theta \in \mathbb{R}$ tais que*

$$U = \mathrm{Scal}(\gamma)\,\mathrm{Ph}(\alpha)\,\mathrm{Rot}(\theta)\,\mathrm{Ph}(\beta).$$

Esses resultados reforçam nossa intuição de que matrizes unitárias representam "rotações gerais" sobre um espaço vetorial. No caso, estamos afirmando que qualquer rotação na esfera de Bloch pode ser decomposta em rotações em eixos individuais. Tais rotações, geradas pelas matrizes de Pauli, são os "blocos fundamentais" de construção de matrizes unitárias em $\mathbb{C}^{2\times 2}$.

## 1.2 Registradores quânticos

Seja $\mathcal{H}_{2^n}$ um espaço de Hilbert de dimensão $2^n$. Denote por $\{0,1\}^n$ o conjunto das cadeias de caracteres de comprimento $n$ sobre o alfabeto $\{0,1\}$ e fixe $B_{2^n} := \big\{|x\rangle \colon x \in \{0,1\}^n\big\}$ uma base ortonormal de $\mathcal{H}_{2^n}$. Por exemplo, para $n = 2$ temos $B_4 = \big\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\big\}$. Um registrador quântico de $n$ qubits é um vetor unitário em $\mathcal{H}_{2^n}$, isto é, um vetor $|\phi\rangle \in \mathcal{H}_{2^n}$ é um registrador quântico de $n$ qubits se

$$|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle, \tag{1.12}$$

com $\alpha_x \in \mathbb{C}$ para todo $x \in \{0,1\}^n$ e

$$\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1. \tag{1.13}$$

A nomenclatura para qubits se estende para os registradores: os estados $|x\rangle$ com $x \in \{0,1\}^n$ são os *estados básicos*, o qubit $|\phi\rangle$ é dito uma *superposição* de estados básicos e o coeficiente complexo $\alpha_x$ é chamado de *amplitude* do estado básico $|x\rangle$ para todo $x \in \{0,1\}^n$.

Será conveniente expressarmos o estado (1.12) como

$$|\phi\rangle = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle, \tag{1.14}$$

onde estamos substituindo as cadeias de caracteres de $\{0,1\}^n$ pelos valores numéricos que essas cadeias representam, se interpretadas como representação binária de números. Por exemplo, para $n = 2$, temos

$$\begin{aligned} |\phi\rangle &= \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \\ &= \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle + \delta|3\rangle. \end{aligned}$$

Continuando a estender a notação de qubits para registradores, a representação na base $B_{2^n}$ do registrador quântico dado por (1.14) é

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2^n-1} \end{pmatrix} = |\phi\rangle = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle.$$

A seguinte questão surge naturalmente: se $|\phi\rangle$ é um registrador cujos qubits, de menos para mais significativos, são $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{n-1}\rangle$, qual é o estado quântico do registrador $|\phi\rangle$, dados os estados de cada um dos qubits? A resposta é: $|\phi\rangle = |\phi_{n-1}\rangle \otimes |\phi_{n-2}\rangle \otimes \cdots \otimes |\phi_0\rangle$, onde $\otimes$ denota o produto tensorial, que definimos a seguir.

Sejam

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \qquad \text{e} \qquad B = \begin{pmatrix} b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{p1} & \cdots & b_{pq} \end{pmatrix}$$

matrizes. O produto tensorial de $A$ e $B$, denotado por $A \otimes B$, é definido como

$$A \otimes B := \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}. \tag{1.15}$$

Assim, um registrador $|\phi\rangle$ cujos qubits são $|\phi_1\rangle$ e $|\phi_0\rangle$, com $|\phi_1\rangle := \alpha_0|0\rangle + \alpha_1|1\rangle$ e $|\phi_0\rangle := \beta_0|0\rangle + \beta_1|1\rangle$, está no estado

$$|\phi\rangle = |\phi_1\rangle \otimes |\phi_0\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \otimes \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}$$

e, portanto,

$$\begin{aligned} |\phi\rangle &= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle \qquad (1.16) \\ &= \alpha_0\beta_0|0\rangle + \alpha_0\beta_1|1\rangle + \alpha_1\beta_0|2\rangle + \alpha_1\beta_1|3\rangle. \end{aligned}$$

Outra questão natural é a inversa da anterior: se $|\phi\rangle$ é um registrador cujos qubits, de menos para mais significativos, são $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{n-1}\rangle$, quais são os estados quânticos de cada um dos qubits, dado o estado do registrador $|\phi\rangle$? Postergamos esta discussão para a seção 1.6.

A medição de um registrador quântico funciona de modo semelhante à medição de qubits: se medirmos um registrador quântico $|\phi\rangle$ dado por (1.14), obtemos o estado básico $|x\rangle$ com probabilidade $|\alpha_x|^2$ e, imediatamente após a medição, o estado do registrador será $|x\rangle$. Ou seja, a superposição que existia anteriormente foi irreversivelmente perdida.

Não é necessário, porém, medirmos todos os qubits do registrador: pode-se medir qubits individuais ou grupos de qubits. Por exemplo, se medirmos apenas o qubit $|\phi_1\rangle$ do registrador $|\phi\rangle$ descrito acima na equação (1.16), existem duas possibilidades de resposta:

- O valor observado é $|0\rangle$. Esse evento ocorre com probabilidade $|\alpha_0\beta_0|^2 + |\alpha_0\beta_1|^2 = |\alpha_0|^2$. Neste caso, o estado do registrador $|\phi\rangle$, imediatamente após a medição, será

  $$|\phi'\rangle = \frac{\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle}{|\alpha_0|^2},$$

  ou seja, projetou-se o estado $|\phi\rangle$ no subespaço gerado por $|00\rangle$ e $|01\rangle$, normalizando-se o resultado para obtermos um vetor unitário.

- O valor observado é $|1\rangle$. Esse evento ocorre com probabilidade $|\alpha_1\beta_0|^2 + |\alpha_1\beta_1|^2 = |\alpha_1|^2$. Neste caso, o estado do registrador $|\phi\rangle$, imediatamente após a medição, será

  $$|\phi'\rangle = \frac{\alpha_1\beta_0|10\rangle + \alpha_1\beta_1|01\rangle}{|\alpha_1|^2},$$

  ou seja, projetou-se o estado $|\phi\rangle$ no subespaço gerado por $|10\rangle$ e $|11\rangle$, normalizando-se o resultado para obtermos um vetor unitário.

Para o caso geral, suponha que seu registrador quântico com $n$ qubits está no estado dado pela equação (1.12) e que você está medindo o qubit $|\phi_j\rangle$, onde o qubit menos significativo é $|\phi_0\rangle$ e o mais significativo é $|\phi_{n-1}\rangle$. Para cada cadeia de caracteres $x \in \{0,1\}^n$, escreva $x = x_{n-1} \cdots x_0$, com $x_k \in \{0,1\}$ para todo $k$. Existem duas possibilidades para o resultado da medição de $|\phi_j\rangle$:

- O valor observado é $|0\rangle$. A probabilidade de ocorrência desse evento é

  $$p_0 := \sum \left\{ |\alpha_x|^2 \colon x \in \{0,1\}^n \text{ e } x_j = 0 \right\}.$$

  Neste caso, o estado do registrador, imediatamente após a medição, será

  $$|\phi'\rangle = \frac{\sum \left\{ \alpha_x |x\rangle \colon x \in \{0,1\}^n \text{ e } x_j = 0 \right\}}{p_0},$$

  onde $p_0$ é simplesmente um fator de normalização.

- O valor observado é $|1\rangle$. A probabilidade de ocorrência desse evento é

  $$p_1 := \sum \left\{ |\alpha_x|^2 \colon x \in \{0,1\}^n \text{ e } x_j = 1 \right\}.$$

  Neste caso, o estado do registrador, imediatamente após a medição, será

  $$|\phi'\rangle = \frac{\sum \left\{ \alpha_x |x\rangle \colon x \in \{0,1\}^n \text{ e } x_j = 1 \right\}}{p_1},$$

  onde $p_1$ é simplesmente um fator de normalização.

O mecanismo de medição de um número arbitrário de qubits de um registrador é análogo. Exibimos apenas um exemplo: suponha que seu registrador quântico $|\phi\rangle$ está no estado dado por (1.12) e que ele tem $n \geq 2$ qubits, com os qubits $|\phi_0\rangle, \ldots, |\phi_{n-1}\rangle$ ordenados como acima. Suponha que os qubits medidos são $|\phi_j\rangle$ e $|\phi_k\rangle$. Existem quatro possibilidades para o resultado de uma medição:

- Os valores observados são ambos $|0\rangle$. A probabilidade de ocorrência desse evento é

  $$p_{00} := \sum \left\{ |\alpha_x|^2 \colon x \in \{0,1\}^n \text{ e } x_j = x_k = 0 \right\}.$$

  Neste caso, o estado do registrador, imediatamente após a medição, será

  $$|\phi'\rangle = \frac{\sum \left\{ \alpha_x |x\rangle \colon x \in \{0,1\}^n \text{ e } x_j = x_k = 0 \right\}}{p_{00}}.$$

- Os valores observados são $|0\rangle$ para $|\phi_j\rangle$ e $|1\rangle$ para $|\phi_k\rangle$. A probabilidade de ocorrência desse evento é

$$p_{01} := \sum \left\{ |\alpha_x|^2 \colon x \in \{0,1\}^n \text{ e } x_j = 0 \text{ e } x_k = 1 \right\}.$$

Neste caso, o estado do registrador, imediatamente após a medição, será

$$|\phi'\rangle = \frac{\sum \left\{ \alpha_x |x\rangle \colon x \in \{0,1\}^n \text{ e } x_j = 0 \text{ e } x_k = 1 \right\}}{p_{01}}.$$

- Os valores observados são $|1\rangle$ para $|\phi_j\rangle$ e $|0\rangle$ para $|\phi_k\rangle$. A probabilidade de ocorrência desse evento é

$$p_{10} := \sum \left\{ |\alpha_x|^2 \colon x \in \{0,1\}^n \text{ e } x_j = 1 \text{ e } x_k = 0 \right\}.$$

Neste caso, o estado do registrador, imediatamente após a medição, será

$$|\phi'\rangle = \frac{\sum \left\{ \alpha_x |x\rangle \colon x \in \{0,1\}^n \text{ e } x_j = 1 \text{ e } x_k = 0 \right\}}{p_{10}}.$$

- Os valores observados são ambos $|1\rangle$. A probabilidade de ocorrência desse evento é

$$p_{11} := \sum \left\{ |\alpha_x|^2 \colon x \in \{0,1\}^n \text{ e } x_j = x_k = 1 \right\}.$$

Neste caso, o estado do registrador, imediatamente após a medição, será

$$|\phi'\rangle = \frac{\sum \left\{ \alpha_x |x\rangle \colon x \in \{0,1\}^n \text{ e } x_j = x_k = 1 \right\}}{p_{11}}.$$

## 1.3 Reversibilidade

Falemos agora de portas quânticas operando sobre mais de um qubit. Uma *porta quântica* sobre $n$ qubits é uma função bijetora de $\mathcal{V}_{2^n}$ para $\mathcal{V}_{2^n}$, onde $\mathcal{V}_{2^n}$ é o conjunto de vetores unitários de $\mathcal{H}_{2^n}$. Em outras palavras, uma porta quântica é uma matriz unitária em $\mathcal{H}_{2^n}$. Essa restrição reflete o fato de que, de acordo com as leis da mecânica quântica, toda transformação num estado quântico é reversível.

Assim, toda porta quântica é reversível, isto é, existe uma bijeção entre o domínio e a imagem da função correspondente. Por exemplo, suponha que temos uma porta quântica $U$ e um registrador $|\phi\rangle$ com dimensões compatíveis. Se aplicarmos $U$ a $|\phi\rangle$ para obtermos $|\phi'\rangle := U|\phi\rangle$, então podemos obter o estado original $|\phi\rangle$ a partir de $|\phi'\rangle$ através da aplicação da porta quântica $U^*$

(claramente $U^*$ é unitária), pois $U^*|\phi'\rangle = U^*U|\phi\rangle = I|\phi\rangle = |\phi\rangle$. Em outras palavras, a aplicação de $U$ não causa "perda de informação".

Esse não é o caso, por exemplo, da porta lógica $\vee$, o "ou" lógico do modelo clássico: suponha que temos dois bits clássicos $a$ e $b$ e que aplicamos a porta $\vee$ nesses bits, obtendo $c := a \vee b$. Se tivermos $c = 1$, não temos como obter os valores de $a$ e $b$, pois podia valer que $a = 1$ e $b = 0$, ou que $a = 0$ e $b = 1$, ou ainda, que $a = 1$ e $b = 1$. Dizemos, por esse motivo, que a porta $\vee$ não é reversível.

Não é difícil, porém, construirmos uma "versão" da porta "ou" que seja reversível. Considere a função $f : \{0,1\}^3 \longrightarrow \{0,1\}^3$ dada por $f(x,y,z) := \big(x, y, z \oplus (x \vee y)\big)$, onde $\oplus$ denota a operação lógica "ou exclusivo". Em outras palavras, a aplicação da função $f$ muda o valor do bit $z$ se, e somente se, $x \vee y = 1$. Além disso, é trivial obtermos os valores originais de $x$, $y$ e $z$, pois $x$ e $y$ fazem parte dos valores que saem da porta. É evidente que $f$ é uma bijeção, de modo que $f$ é reversível. A única diferença é que estamos armazenando informações a mais, o suficiente para sermos capazes de obter $x$ e $y$ (e $z$) a partir de $f(x,y,z)$.

Então existe uma matriz unitária $U_f$ que "implementa" a função $f$. Dado um registrador $|\phi\rangle := |x\rangle \otimes |y\rangle \otimes |z\rangle$, onde $|x\rangle, |y\rangle$ e $|z\rangle$ são qubits, a porta $U_f$ leva $|\phi\rangle$ ao estado $|\phi'\rangle = |x\rangle \otimes |y\rangle \otimes \big|z \oplus (x \vee y)\big\rangle$. Será mais conveniente usarmos a seguinte notação: dado um registrador $|\phi\rangle := |x, y, z\rangle$, a aplicação de $U_f$ a $|\phi\rangle$ leva o estado deste registrador a $|\phi'\rangle = \big|x, y, z \oplus (x \vee y)\big\rangle$. Veja que estamos simplesmente separando os qubits individuais por vírgulas. Na verdade, podemos utilizar essa notação para separar grupos arbitrários de qubits num registrador, quando isso for conveniente.

O "truque" de carregar informações a mais nas aplicações de portas lógicas pode ser utilizado para transformar qualquer porta lógica do modelo clássico numa porta reversível e que, portanto, pode ser implementada por uma matriz unitária no modelo quântico.

Na verdade, Bennett [Ben73] mostrou que qualquer circuito no modelo clássico pode ser convertido em reversível, ou seja, num circuito cujas portas são todas reversíveis, e mostrou que isso pode ser feito com um aumento no máximo polinomial no tamanho do circuito, isto é, no número de portas utilizadas. Para mais detalhes, consulte o apêndice E.

## 1.4 Circuitos quânticos

A notação para circuitos apresentada na seção sobre qubits se estende facilmente para circuitos operando sobre múltiplos qubits. Por exemplo, a figura 1.5

mostra um circuito operando sobre 3 qubits e com uma única porta, dada pela matriz $U_f$ implementando a versão reversível da operação lógica "ou". De acordo com nossa especificação de $U_f$, temos $|x'\rangle = |x\rangle$, $|y'\rangle = |y\rangle$ e $|z'\rangle = \big|z \oplus (x \vee y)\big\rangle$.

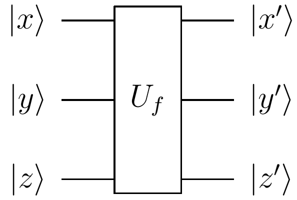


Figura 1.5: Circuito com porta "ou" reversível.

Considere agora o circuito ilustrado na figura 1.6. Veremos tal circuito mais adiante quando apresentarmos o problema de Deutsch. Por enquanto, vamos apenas analisar o estado do registrador a cada aplicação de porta. Considere o registrador $|\phi\rangle$ de 2 qubits formado pelos qubits $|x\rangle$ e $|y\rangle$. Suponha que o estado inicial de $|\phi\rangle$, antes da aplicação do circuito, é $|\phi\rangle = |x\rangle \otimes |y\rangle$.

Após a aplicação da primeira "camada" de portas do circuito, isto é, após a aplicação de $H$ a cada um dos qubits, o estado do registrador será $|\phi'\rangle := \big(H|x\rangle\big) \otimes \big(H|y\rangle\big) = (H \otimes H)\big(|x\rangle \otimes |y\rangle\big) = (H \otimes H)|\phi\rangle$, pelas propriedades do produto tensorial. Podemos resumir isso informalmente: se numa dada "camada" do circuito tivermos várias portas, cada uma operando sobre um certo conjunto de qubits, então a transformação unitária aplicada ao registrador é dada pelo produto tensorial de cada uma das portas.

Agora aplicamos a segunda "camada" de portas ao registrador, obtendo o estado $|\phi''\rangle := U_f|\phi'\rangle$. Finalmente, após a aplicação da terceira "camada", isto é, após a aplicação de $H$ ao qubit $|x\rangle$, o estado do registrador será $|\phi'''\rangle := (H \otimes I)|\phi''\rangle$. Resumindo: se, numa camada, nenhuma porta opera sobre um dado qubit, então o termo do produto tensorial referente a tal qubit é a matriz identidade.

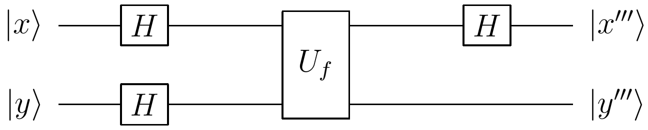


Figura 1.6: Circuito para o problema de Deutsch.

Seja $V_f \in \mathbb{C}^{2\times 2}$ uma matriz unitária. Então a notação utilizada no circuito da figura 1.7 indica que o operador $V_f$ deve ser aplicado ao qubit $|y\rangle$ se, e somente se, $|x\rangle = |1\rangle$. Em outras palavras, o circuito da figura 1.7 é equivalente ao circuito da figura 1.8, onde

$$V'_f := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & v_{11} & v_{12} \\ 0 & 0 & v_{21} & v_{22} \end{pmatrix} \qquad \text{e} \qquad V_f := \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix}.$$

A porta indicada no circuito da figura 1.7 é chamada *porta $V_f$ controlada por $|x\rangle$*.

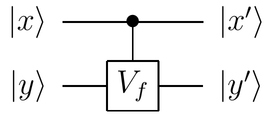


Figura 1.7: Circuito com porta $V_f$ controlada por $|x\rangle$.

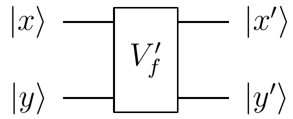


Figura 1.8: Circuito equivalente ao da figura 1.7.

## 1.5 Teorema do no-cloning

Não existe uma transformação que copia (clona) o estado de qualquer qubit para outro. Note que, se isso fosse possível, poderíamos copiar o estado desse qubit antes de realizarmos uma medição, de modo que teríamos uma medição do qubit e ainda assim teríamos uma cópia intacta. Mais que isso, se tivéssemos um número ilimitado de cópias de um qubit desconhecido, poderíamos determinar seu estado quântico com precisão arbitrária.

**Teorema 1.6 (No-cloning).** *Não existe uma transformação unitária $U$ tal que $U|\psi, 0\rangle = |\psi, \psi\rangle$ para qualquer qubit $|\psi\rangle$.*

*Demonstração.* Suponha que tal $U$ existe. Sejam $|\alpha\rangle, |\beta\rangle$ qubits ortogonais. Temos $U|\alpha, 0\rangle = |\alpha, \alpha\rangle$ e $U|\beta, 0\rangle = |\beta, \beta\rangle$. Seja $|\gamma\rangle = \big(|\alpha\rangle + |\beta\rangle\big)/\sqrt{2}$. Então

$$\begin{aligned} U|\gamma, 0\rangle &= U\left[\frac{1}{\sqrt{2}}\big(|\alpha, 0\rangle + |\beta, 0\rangle\big)\right] = \frac{1}{\sqrt{2}}\big(U|\alpha, 0\rangle + U|\beta, 0\rangle\big) \\ &= \frac{1}{\sqrt{2}}\big(|\alpha, \alpha\rangle + |\beta, \beta\rangle\big). \end{aligned}$$

Porém, temos também $U|\gamma,0\rangle = |\gamma,\gamma\rangle = \frac{1}{2}\big(|\alpha,\alpha\rangle + |\alpha,\beta\rangle + |\beta,\alpha\rangle + |\beta,\beta\rangle\big)$.

Como $|\alpha\rangle$ e $|\beta\rangle$ são vetores ortogonais, os elementos do conjunto $B := \big\{|\alpha,\alpha\rangle, |\alpha,\beta\rangle, |\beta,\alpha\rangle, |\beta,\beta\rangle\big\}$ são mutuamente ortogonais (lembre-se que $\langle\phi,\psi|\phi',\psi'\rangle = \langle\phi|\phi'\rangle\langle\psi|\psi'\rangle$) e então $B$ é uma base para $\mathcal{H}_4$, de modo que todo vetor deste espaço pode ser escrito de uma única forma como combinação linear dos elementos de $B$. Mas então

$$\begin{aligned} U|\gamma,0\rangle &= \frac{1}{\sqrt{2}}\big(|\alpha,\alpha\rangle + |\beta,\beta\rangle\big) \\ &\neq \frac{1}{2}\big(|\alpha,\alpha\rangle + |\alpha,\beta\rangle + |\beta,\alpha\rangle + |\beta,\beta\rangle\big) = U|\gamma,0\rangle, \end{aligned}$$

uma contradição. □

Vamos provar uma limitação acerca de transformações "de cópia":

**Proposição 1.7.** *Sejam $U$ uma transformação unitária e $|\psi\rangle, |\phi\rangle$ qubits distintos. Se $U|\psi,0\rangle = |\psi,\psi\rangle$ e $U|\phi,0\rangle = |\phi,\phi\rangle$, então $\langle\psi|\phi\rangle = 0$.*

*Demonstração.* Primeiro notamos que $\langle\psi,\psi|\phi,\phi\rangle = \langle\psi|\phi\rangle^2$. Além disso, temos

$$\begin{aligned} \langle\psi,\psi|\phi,\phi\rangle &= \big[U|\psi,0\rangle\big]^*\big[U|\phi,0\rangle\big] = \big[|\psi,0\rangle^* U^*\big]\big[U|\phi,0\rangle\big] = |\psi,0\rangle^*|\phi,0\rangle \\ &= \langle\psi,0|\phi,0\rangle = \langle\psi|\phi\rangle\langle 0|0\rangle = \langle\psi|\phi\rangle. \end{aligned}$$

Com isso, $\langle\psi|\phi\rangle^2 = \langle\psi|\phi\rangle$ e portanto $\langle\psi|\phi\rangle$ só pode ser 0 ou 1. Porém, como $\|\psi\| = \|\phi\| = 1$, o produto interno de $|\psi\rangle$ e $|\phi\rangle$ só pode ser 1 se $|\psi\rangle = |\phi\rangle$. Como esse não é o caso, segue que $\langle\psi|\phi\rangle = 0$. □

É importante observar que tipo de cópia o teorema do no-cloning proíbe. Por exemplo, é possível clonar um estado quântico conhecido. O que o teorema nos afirma é que é impossível que uma transformação copie um estado quântico arbitrário.

**Proposição 1.8.** *Seja $|\psi\rangle$ um qubit. Então existe uma operação unitária $U_{|\psi\rangle}$ tal que $U_{|\psi\rangle}|\psi,0\rangle = |\psi,\psi\rangle$.*

*Demonstração.* Seja $|\psi\rangle := \alpha|0\rangle + \beta|1\rangle$. É fácil ver que

$$U_{|\psi\rangle} = I \otimes \begin{pmatrix} \alpha & -\beta^* \\ \beta & \alpha^* \end{pmatrix}$$

é unitária e satisfaz a afirmação. □

## 1.6 Emaranhamento quântico

Considere um registrador de 2 qubits $|\psi\rangle$ no estado

$$|\Phi^+\rangle := \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big).$$

Se fizermos uma medição do primeiro qubit em relação à base padrão, obteremos $|0\rangle$ com probabilidade $1/2$ e $|1\rangle$ com probabilidade $1/2$. No primeiro caso, $|\psi\rangle$ passa a valer $|00\rangle$ e, no segundo, $|11\rangle$. Com isso, uma posterior medição no segundo qubit já tem seu resultado determinado, isto é, com probabilidade 1 obteremos $|0\rangle$ no primeiro caso e $|1\rangle$ no segundo. Os qubits neste estado não são independentes ou não têm identidade própria.

Isso não ocorre, entretanto, se $|\psi\rangle$ estivesse no estado

$$\frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\big),$$

isto é, o resultado de uma medição do primeiro qubit não afeta o resultado de medições no segundo qubit realizadas posteriormente.

Em geral, se o estado de $|\psi\rangle$ puder ser escrito como o produto tensorial de dois qubits, então estes "têm identidade própria". Suponha que $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, onde $|\psi_1\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ e $|\psi_2\rangle = \beta_0|0\rangle + \beta_1|1\rangle$ são qubits. Se medirmos o primeiro qubit com relação à base padrão, obtemos $|0\rangle$ com probabilidade $|\alpha_0\beta_0|^2 + |\alpha_0\beta_1|^2 = |\alpha_0|^2$ e o novo estado de $|\psi\rangle$ será $|0\rangle \otimes |\psi_2\rangle$; obtemos $|1\rangle$ com probabilidade $|\alpha_1\beta_0|^2 + |\alpha_1\beta_1|^2 = |\alpha_1|^2$ e $|\psi\rangle$ passa a ser $|1\rangle \otimes |\psi_2\rangle$.

Já se $|\psi\rangle$ não puder ser escrito como produto tensorial de dois qubits, eles demonstram um comportamento semelhante ao caso de $|\Phi^+\rangle$ mostrado acima. Note que $|\Phi^+\rangle$ não pode ser escrito como produto tensorial de dois qubits.

Seja $|\psi\rangle$ um registrador com $n$ qubits, isto é, $|\psi\rangle \in \mathcal{H}_{2^n} := \bigotimes_{i=1}^n \mathcal{H}_2$. Se $|\psi\rangle$ não pode ser escrito na forma $|\psi\rangle = \bigotimes_{i=1}^n |\psi_i\rangle$, onde $|\psi_i\rangle$ é um qubit para $1 \leq i \leq n$, então $|\psi\rangle$ é chamado de *estado emaranhado* (*entangled state*).

**Exemplo 1.9** Tome a seguinte matriz de dimensão $2^n$, para $n > 1$:

$$E := \frac{1}{\sqrt{2}}\begin{pmatrix} I & S \\ S & -I \end{pmatrix},$$

onde $I$ é a matriz identidade de dimensão $2^{n-1}$ e $S$ é a matriz de mesma dimensão contendo 0's em todas as entradas exceto na diagonal secundária, onde $S$ tem 1's.

É fácil ver que $E$ é unitária:

$$
\begin{aligned}
E^*E &= \frac{1}{2}\begin{pmatrix} I^* & S^* \\ S^* & -I^* \end{pmatrix}\begin{pmatrix} I & S \\ S & -I \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} I+S^2 & 0 \\ 0 & I+S^2 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 2I & 0 \\ 0 & 2I \end{pmatrix},
\end{aligned}
$$

que é justamente a matriz identidade de dimensão $2^n$.

Agora observe que $E$, aplicada no vetor básico $|0^n\rangle$, gera o estado emaranhado $\frac{1}{\sqrt{2}}\big(|0^n\rangle + |1^n\rangle\big)$. Já aplicada em $|1^n\rangle$, gera o estado emaranhado $\frac{1}{\sqrt{2}}\big(|0^n\rangle - |1^n\rangle\big)$.

Um registrador de 2 qubits no estado $|\Phi^+\rangle$ é dito um estado EPR, ou par EPR, onde EPR é uma abreviação para Einstein, Podolsky e Rosen, físicos que contestaram a validade desta formulação da teoria quântica, já que esta possibilitava situações aparentemente paradoxais, que violam princípios fundamentais da teoria da relatividade.

A existência de estados quânticos emaranhados é a principal razão pela qual computadores quânticos não podem ser facilmente simulados eficientemente por computadores clássicos (mesmo no modelo probabilístico). De fato, se todo registrador quântico com $n$ qubits pudesse sempre ser escrito como o produto tensorial de $n$ qubits, então bastaria armazenar classicamente $2n$ números complexos por registrador quântico. Entretanto, é uma conseqüência da existência de estados emaranhados que nem sempre podemos dividir um estado quântico em suas "partes componentes" e operar nestas separadamente. Assim, para armazenar classicamente o estado de um registrador quântico de $n$ qubits, parece ser necessário armazenar $2^n$ números complexos.

# Capítulo 2

# Algoritmos quânticos

Estudaremos agora conceitos fundamentais sobre algoritmos no modelo quântico. Começamos mostrando o paralelismo quântico, técnica essencial para o projeto de algoritmos eficientes. A seguir, formalizamos o conceito de medida de tempo consumido por um algoritmo neste novo modelo de computação. Finalmente, apresentamos alguns exemplos simples de algoritmos, como os que resolvem os problemas de Deutsch, de Deutsch-Jozsa e de Simon.

## 2.1 Paralelismo quântico

Considere um registrador com $n$ qubits no estado

$$|\phi\rangle := \sum_{i=0}^{2^n-1} \alpha_i |i\rangle.$$

A aplicação de uma matriz unitária $U$ de dimensão $2^n$ sobre este registrador produz

$$|\phi'\rangle = U|\phi\rangle = U\left(\sum_{i=0}^{2^n-1} \alpha_i |i\rangle\right) = \sum_{i=0}^{2^n-1} \alpha_i U|i\rangle,$$

isto é, uma única aplicação de $U$ realiza um número exponencial de operações em estados básicos. Este fenômeno é chamado de *paralelismo quântico (quantum parallelism)*.

Usualmente, a matriz de Hadamard, dada por (1.3), é utilizada para gerar num registrador um estado quântico contendo a superposição de todos os estados básicos, todos com a mesma amplitude.

Seja $H_n := \bigotimes_{j=1}^{n} H$, onde $H$ é a matriz de Hadamard, como definimos em (1.3). Chame $H_n$ de matriz de Hadamard de ordem $2^n$. Note que a aplicação de $H_n$ a um registrador é feita simplesmente pela aplicação de $H$ a cada qubit individual do registrador. Abaixo ilustramos o uso dessa transformação.

**Exemplo 2.1** Seja $f : \{0, \ldots, 2^n - 1\} \longrightarrow \{0, \ldots, 2^m - 1\}$ uma função. Considere um registrador $|\phi\rangle$ com $n+m$ qubits, formado por dois sub-registradores, um com $n$ qubits contendo $|a\rangle$, para algum $0 \leq a < 2^n$, e outro com $m$ qubits contendo $|b\rangle$, para algum $0 \leq b < 2^m$, ou seja, $|\phi\rangle$ é o produto tensorial de $|a\rangle$ e $|b\rangle$: $|\phi\rangle = |a, b\rangle$. Seja $U_f$ uma matriz unitária de dimensão $2^{n+m}$ tal que $U_f|a, b\rangle = \big|a, b \oplus f(a)\big\rangle$.

Denote por $I_m$ a matriz identidade de ordem $2^m$. Partindo do registrador $|\phi\rangle$ inicializado com o estado básico $|0, 0\rangle$, aplicamos $H_n$ sobre o primeiro sub-registrador, isto é, aplicamos o operador $H_n \otimes I_m$ sobre o registrador $|\phi\rangle$, obtendo

$$(H_n \otimes I_m)|0, 0\rangle = \big(H_n|0\rangle\big) \otimes \big(I_m|0\rangle\big).$$

Não é difícil ver que

$$\begin{aligned} H_n|0^n\rangle &= \left(\bigotimes_{j=1}^{n} H\right)\left(\bigotimes_{j=1}^{n} |0\rangle\right) = \bigotimes_{j=1}^{n} H|0\rangle \\ &= \frac{1}{\sqrt{2^n}} \bigotimes_{j=1}^{n} \big(|0\rangle + |1\rangle\big) = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle. \end{aligned} \tag{2.1}$$

Então

$$(H_n \otimes I_m)|0, 0\rangle = \left(\frac{1}{\sqrt{2}} \sum_{x=0}^{2^n-1} |x\rangle\right) \otimes |0\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{2^n-1} |x, 0\rangle.$$

Agora aplicamos $U_f$ a $|\phi\rangle$ uma única vez, para obter

$$\begin{aligned} U_f\left(\frac{1}{\sqrt{2}} \sum_{x=0}^{2^n-1} |x, 0\rangle\right) &= \frac{1}{\sqrt{2}} \sum_{x=0}^{2^n-1} U_f|x, 0\rangle \\ &= \frac{1}{\sqrt{2}} \sum_{x=0}^{2^n-1} \big|x, 0 \oplus f(x)\big\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{2^n-1} \big|x, f(x)\big\rangle. \end{aligned}$$

Observe que, com uma única aplicação de $U_f$ o valor de $f(x)$ foi calculado para todo $0 \leq x < 2^n$.

Vale lembrar que a medição de um registrador como descrito acima causa o colapso deste em um único estado básico, justamente o que é obtido na medição.

Desta forma, apesar de o estado do registrador conter o valor de $f(x)$ para todo $0 \leq x < 2^m$, podemos obter um único valor da função. Além disso, não podemos nem escolher para qual $x$ obteremos $f(x)$, já que o resultado da medição é probabilístico. É preciso então desenvolver operações unitárias que manipulem de forma inteligente as amplitudes dos estados básicos para que se possa usar eficientemente o paralelismo quântico.

**Exemplo 2.2 (Mudança de sinal)** Sejam $f$ e $U_f$ como no exemplo acima e tome $m = 1$. Seja $|x\rangle$ um registrador com $n$ qubits. Vamos mostrar como realizar a operação $V_f$ dada por

$$V_f|x\rangle = (-1)^{f(x)}|x\rangle,$$

que muda o sinal da amplitude dos estados básicos $|x\rangle$ para os quais $f(x) = 1$.

Para tanto, utilizaremos um qubit adicional, no estado $\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)$, que é facilmente obtido através da aplicação da transformação de Hadamard ao estado básico $|1\rangle$. Este qubit é "concatenado" ao registrador $|x\rangle$ através do produto tensorial, formando

$$|x\rangle \otimes \left[\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)\right] = \frac{1}{\sqrt{2}}\big(|x,0\rangle - |x,1\rangle\big).$$

Aplicando $U_f$ a este estado obtemos

$$U_f\left[\frac{1}{\sqrt{2}}\big(|x,0\rangle - |x,1\rangle\big)\right] = \frac{1}{\sqrt{2}}\Big[\big|x, f(x)\big\rangle - \big|x, 1 \oplus f(x)\big\rangle\Big].$$

Se $f(x) = 0$, o estado anterior é

$$\begin{aligned} \frac{1}{\sqrt{2}}\Big[|x,0\rangle - |x,1 \oplus 0\rangle\Big] &= \frac{1}{\sqrt{2}}\big(|x,0\rangle - |x,1\rangle\big) \\ &= (-1)^0\left\{|x\rangle \otimes \left[\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)\right]\right\}. \end{aligned}$$

Já se $f(x) = 1$,

$$\begin{aligned} \frac{1}{\sqrt{2}}\Big[|x,1\rangle - |x,1 \oplus 1\rangle\Big] &= -\frac{1}{\sqrt{2}}\big(|x,0\rangle - |x,1\rangle\big) \\ &= (-1)^1\left\{|x\rangle \otimes \left[\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)\right]\right\}. \end{aligned}$$

Ou seja,

$$U_f\left\{|x\rangle \otimes \left[\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)\right]\right\} = (-1)^{f(x)}\left\{|x\rangle \otimes \left[\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)\right]\right\}, \qquad (2.2)$$

como queríamos.

## 2.2 Medida do consumo de tempo

Para medirmos o consumo de tempo de um algoritmo no modelo quântico, vamos analisar o circuito equivalente à sua execução para cada instância possível. Essa formulação baseada em circuitos é igualmente válida para o modelo clássico. Mais adiante discutimos alguns aspectos de precisão que surgem apenas no modelo quântico.

Suponha que um algoritmo $\mathcal{A}$ resolve um certo problema e seja $I$ uma instância desse problema. Seja $n$ o tamanho de $I$, isto é, suponha que $I$ pode ser codificada com $n$ qubits. A execução de $\mathcal{A}$ sobre $I$ equivale a um circuito $\mathcal{C}$ que tem os qubits de $I$ como entrada. Precisamos fazer algumas exigências acerca de $\mathcal{C}$. Fixe um inteiro $k \geq 2$. Exigimos que:

(i) dado $I$, exista um algoritmo polinomial em $n$ capaz de construir o circuito $\mathcal{C}$ equivalente à execução de $\mathcal{A}$ sobre $I$;

(ii) cada porta do circuito $\mathcal{C}$ opere sobre, no máximo, $k$ qubits.

Satisfeitas essas exigências, diremos que o consumo de tempo do algoritmo $\mathcal{A}$ para a instância $I$ é o tamanho de $\mathcal{C}$, isto é, o número de portas do circuito.

A exigência (i) apenas impede que o algoritmo usado para construir o circuito $\mathcal{C}$ utilize tempo superpolinomial. De fato, se não impuséssemos limitações para o consumo de tempo desse algoritmo de construção, o tamanho do circuito $\mathcal{C}$ poderia ser até mesmo constante.

Já a exigência (ii) garante que cada passo da computação é feito localmente. Em outras palavras, cada porta do circuito opera sobre $O(1)$ qubits. Como o consumo de tempo do algoritmo é o número de portas do circuito, estamos simplesmente dizendo que cada passo do algoritmo "custa" $O(1)$.

Tudo que foi discutido até aqui pode ser aplicado ao modelo clássico. Mas, no modelo quântico, pode surgir a seguinte questão: dentre todas as portas que operam sobre no máximo $k$ qubits, quais delas podemos utilizar no circuito, ou seja, quais delas são fisicamente realizáveis? Esse problema não aparece no modelo clássico pois o número de portas que opera sobre no máximo $k$ bits é finito. Porém, no modelo quântico, o número de portas operando sobre no máximo $k$ qubits não só é infinito, como também é não-enumerável.

A resposta para essa questão vem da existência de portas quânticas universais. Em outras palavras, existe um conjunto de portas quânticas, que chamaremos de família universal de portas quânticas, capaz de "simular" a aplicação de qualquer matriz unitária. Vamos formalizar melhor esse conceito.

Seja $U$ uma matriz unitária operando sobre no máximo $k$ qubits. Ou seja, a dimensão de $U$ é $m := 2^l$, com $l \leq k$. Então para todo $\epsilon > 0$ existe um circuito quântico, utilizando somente as portas de uma família universal de portas quânticas, cuja aplicação equivale à transformação unitária $U'$, sendo que $\|U - U'\| < \epsilon$. Ademais, tal circuito tem tamanho polinomial em $1/\epsilon$.

Em outras palavras, se tivermos em mãos um computador quântico equipado com uma família universal de portas quânticas, podemos simular com precisão arbitrária qualquer transformação unitária operando sobre no máximo $k$ qubits. Não vamos considerar a influência de erros a cada passo do algoritmo. Para mais detalhes, veja as notas de aula de Preskill [Pre04] e o artigo de Bernstein e Vazirani [BV97].

Portanto, podemos utilizar qualquer porta quântica operando sobre no máximo $k$ qubits.

## 2.3 O problema de Deutsch

Dizemos que uma função $f$ é dada como uma *caixa preta* se só podemos obter informações acerca de $f$ através de sua aplicação a elementos de seu domínio. O problema de Deutsch, também conhecido como problema XOR de Deutsch, consiste no seguinte:

**Problema 2.3 (XOR de Deutsch)** Seja $f : \{0,1\} \longrightarrow \{0,1\}$ uma função dada como uma caixa preta. Determine se $f(0) = f(1)$ ou se $f(0) \neq f(1)$.

Se $f(0) = f(1)$, dizemos que $f$ é *constante*. Já se $f(0) \neq f(1)$, dizemos que $f$ é *balanceada*.

Para se resolver o problema com certeza no modelo clássico, são necessárias duas aplicações de $f$: é preciso usar a caixa preta de $f$ duas vezes, para as entradas 0 e 1. Já no modelo quântico, este problema pode ser resolvido satisfatoriamente utilizando-se apenas uma chamada à caixa preta. Primeiro mostramos a solução original dada por Deutsch, que dá a resposta certa com probabilidade $1/2$ e não dá resposta alguma com probabilidade $1/2$. Depois mostramos uma solução melhorada, dada por Cleve, Ekert, Macchiavello e Mosca [CEMM98], que sempre dá a resposta certa.

**Solução 2.4 (Deutsch)** Seja $U_f$ a transformação unitária de dimensão 4 que leva $|x,y\rangle$ a $\big|x, y \oplus f(x)\big\rangle$. No modelo quântico, $U_f$ é a nossa caixa preta. Tome um registrador $|\phi_0\rangle$ com 2 qubits inicializado com $|0,0\rangle$ e aplique a transformação de Hadamard no primeiro qubit, obtendo

$$\begin{aligned} |\phi_1\rangle &:= \big(H \otimes I\big)\big(|0\rangle \otimes |0\rangle\big) \\ &= \left[\frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big)\right] \otimes |0\rangle \;=\; \frac{1}{\sqrt{2}}\Big(|0,0\rangle + |1,0\rangle\Big). \end{aligned}$$

Com uma única aplicação da caixa preta $U_f$ a $|\phi_1\rangle$, obtemos o estado

$$|\phi_2\rangle := U_f|\phi_1\rangle = \frac{1}{\sqrt{2}}\Big(\big|0, f(0)\big\rangle + \big|1, f(1)\big\rangle\Big).$$

Aplicamos agora $H_2$ sobre $|\phi_2\rangle$ para obter $|\phi_3\rangle$. Lembrando que

$$H_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix},$$

temos

$$\begin{aligned}|\phi_3\rangle = \frac{1}{\sqrt{2}}\Bigg[&\frac{1}{2}\Big(|0,0\rangle + (-1)^{f(0)}|0,1\rangle + |1,0\rangle + (-1)^{f(0)}|1,1\rangle\Big) \\ &+ \frac{1}{2}\Big(|0,0\rangle + (-1)^{f(1)}|0,1\rangle - |1,0\rangle + (-1)^{f(1)+1}|1,1\rangle\Big)\Bigg],\end{aligned}$$

e portanto

$$\begin{aligned}|\phi_3\rangle = \frac{1}{\sqrt{2}}\Bigg[&|0,0\rangle + \frac{1}{2}\Big((-1)^{f(0)} + (-1)^{f(1)}\Big)|0,1\rangle \\ &+ \frac{1}{2}\Big((-1)^{f(0)} + (-1)^{f(1)+1}\Big)|1,1\rangle\Bigg].\end{aligned}$$

Assim, se $f$ é constante, então $f(0) = f(1)$ e $f(0) \not\equiv f(1) + 1 \pmod 2$, de modo que

$$|\phi_3\rangle = \frac{1}{\sqrt{2}}\Big(|0,0\rangle + (-1)^{f(0)}|0,1\rangle\Big).$$

Já se $f$ é balanceada, então $f(0) \not\equiv f(1) \pmod 2$ e $f(0) \equiv f(1) + 1 \pmod 2$, de forma que

$$|\phi_3\rangle = \frac{1}{\sqrt{2}}\Big(|0,0\rangle + (-1)^{f(0)}|1,1\rangle\Big).$$

Fazemos agora uma medição do segundo qubit. Se obtivermos $|0\rangle$, perdemos nossos cálculos. Mas com probabilidade $1/2$ obteremos $|1\rangle$. Neste caso, ocorre um colapso no registrador, que estará no estado $|0,1\rangle$ se $f$ é constante ou no estado $|1,1\rangle$ se $f$ for balanceada. Assim, uma medição do primeiro qubit nos fornece o resultado correto.

**Solução 2.5 (Cleve, Ekert, Macchiavello e Mosca)** Considere novamente a transformação $U_f$ como nossa caixa preta e um registrador $|\phi_0\rangle$ com 2 qubits inicializado com $|0,1\rangle$.

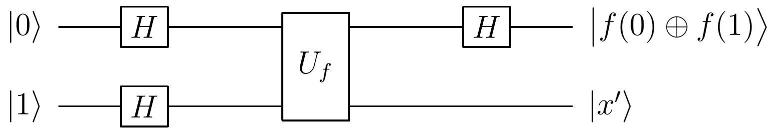


Figura 2.1: Circuito para o problema de Deutsch

O circuito quântico ilustrado na figura 2.1 mostra o algoritmo que resolve o problema de Deutsch. Acompanhe a seguir o funcionamento do algoritmo:

Primeiro aplicamos a transformação de Hadamard aos 2 qubits do registrador, obtendo

$$\begin{aligned} |\phi_1\rangle &:= H_2|\phi_0\rangle = \big(H|0\rangle\big)\otimes\big(H|1\rangle\big) = \frac{1}{2}\Big[\Big(|0\rangle+|1\rangle\Big)\otimes\Big(|0\rangle-|1\rangle\Big)\Big] \\ &= \frac{1}{2}\Big[|0\rangle\otimes\Big(|0\rangle-|1\rangle\Big)\Big]+\frac{1}{2}\Big[|1\rangle\otimes\Big(|0\rangle-|1\rangle\Big)\Big]. \end{aligned}$$

Neste ponto a caixa preta $U_f$ é aplicada a $|\phi_1\rangle$ para obtermos $|\phi_2\rangle$:

$$|\phi_2\rangle := U_f|\phi_1\rangle = \frac{1}{2}\Bigg\{U_f\Big[|0\rangle\otimes\Big(|0\rangle-|1\rangle\Big)\Big]\Bigg\}+\frac{1}{2}\Bigg\{U_f\Big[|1\rangle\otimes\Big(|0\rangle-|1\rangle\Big)\Big]\Bigg\}.$$

Já vimos em (2.2) do exemplo da página 32 que a aplicação de $U_f$ a um registrador contendo $|\phi\rangle = |x\rangle\otimes\big(|0\rangle-|1\rangle\big)$, onde $x\in\{0,1\}$, resulta em $(-1)^{f(x)}|\phi\rangle$. Assim,

$$\begin{aligned} |\phi_2\rangle &= \frac{1}{2}\Bigg\{(-1)^{f(0)}\Big[|0\rangle\otimes\Big(|0\rangle-|1\rangle\Big)\Big]\Bigg\}+\frac{1}{2}\Bigg\{(-1)^{f(1)}\Big[|1\rangle\otimes\Big(|0\rangle-|1\rangle\Big)\Big]\Bigg\} \\ &= \frac{1}{2}\Big[\Big((-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle\Big)\otimes\Big(|0\rangle-|1\rangle\Big)\Big] \\ &= \frac{1}{2}\Big[\Big((-1)^{f(0)}|0\rangle+(-1)^{f(1)}(-1)^{f(0)}(-1)^{f(0)}|1\rangle\Big)\otimes\Big(|0\rangle-|1\rangle\Big)\Big] \\ &= \frac{(-1)^{f(0)}}{2}\Big[\Big(|0\rangle+(-1)^{f(0)\oplus f(1)}|1\rangle\Big)\otimes\Big(|0\rangle-|1\rangle\Big)\Big] \\ &= (-1)^{f(0)}\Bigg\{\Big[\frac{1}{\sqrt{2}}\Big(|0\rangle+(-1)^{f(0)\oplus f(1)}|1\rangle\Big)\Big]\otimes\Big[\frac{1}{\sqrt{2}}\Big(|0\rangle-|1\rangle\Big)\Big]\Bigg\}. \end{aligned}$$

Agora podemos aplicar a transformação de Hadamard ao primeiro qubit de $|\phi_2\rangle$ para obter

$$|\phi_3\rangle := (H\otimes I)|\phi_2\rangle = (-1)^{f(0)}\Bigg\{\Big[\big|f(0)\oplus f(1)\big\rangle\Big]\otimes\Big[\frac{1}{\sqrt{2}}\Big(|0\rangle-|1\rangle\Big)\Big]\Bigg\}.$$

Uma medição do primeiro qubit de $|\phi_3\rangle$ fornece agora o valor de $f(0) \oplus f(1)$ e portanto o algoritmo descobre com certeza se $f$ é constante ou balanceada através de uma única aplicação da caixa preta.

## 2.4 O problema de Deutsch-Jozsa

Deutsch e Jozsa propuseram mais tarde uma generalização do problema de Deutsch, que descrevemos a seguir.

Seja $f : \{0,1\}^n \longrightarrow \{0,1\}$ uma função. Dizemos que $f$ é *constante* se $f(x) = f(y)$ para todos $x, y \in \{0,1\}^n$ e que $f$ é *balanceada* se $|F_0| = |F_1|$, onde $F_z := \big\{x \in \{0,1\}^n : f(x) = z\big\}$ para $z = 0, 1$.

O problema de Deutsch-Jozsa consiste no seguinte:

**Problema 2.6 (Deutsch-Jozsa)** Seja $f : \{0,1\}^n \longrightarrow \{0,1\}$ uma função dada como uma caixa preta, com a garantia de que $f$ é constante ou balanceada. Determine se $f$ é constante ou balanceada.

A solução original proposta por Deutsch e Jozsa faz duas chamadas à caixa preta $U_f$, que é a transformação que leva $|x, y\rangle$ a $\big|x, y \oplus f(x)\big\rangle$. Vamos apresentar um algoritmo devido a Cleve, Ekert, Macchiavello e Mosca [CEMM98], que faz apenas uma chamada à caixa preta $U_f$ para resolver o problema. O algoritmo segue de perto a solução exata do problema XOR.

Começamos com um registrador $|\phi_0\rangle$ de $n+1$ qubits inicializado com $|0^n, 1\rangle$, ou seja, os $n$ primeiros qubits valem $|0\rangle$ e apenas o último tem o valor $|1\rangle$. Aplicamos a transformação de Hadamard a cada um dos $n+1$ qubits de $|\phi_0\rangle$, obtendo

$$
\begin{aligned}
|\phi_1\rangle &:= (H_n \otimes H)|\phi_0\rangle = \big(H_n|0\rangle\big) \otimes \big(H|1\rangle\big) \\
&= \left[\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle\right] \otimes \Big(|0\rangle - |1\rangle\Big) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \Big[|x\rangle \otimes \Big(|0\rangle - |1\rangle\Big)\Big].
\end{aligned}
$$

Agora aplicamos a caixa preta $U_f$ a $|\phi_1\rangle$:

$$
\begin{aligned}
|\phi_2\rangle &:= U_f|\phi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} U_f\Big[|x\rangle \otimes \Big(|0\rangle - |1\rangle\Big)\Big] \\
&= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}\Big[|x\rangle \otimes \Big(|0\rangle - |1\rangle\Big)\Big].
\end{aligned}
$$

Observe que novamente utilizamos o que foi mostrado em (2.2) no exemplo da página 32.

Temos assim

$$\begin{aligned}
|\phi_2\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \Big[ |x\rangle \otimes \Big( |0\rangle - |1\rangle \Big) \Big] \\
&= \left[ \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \right] \otimes \Big( |0\rangle - |1\rangle \Big).
\end{aligned}$$

A partir deste ponto passamos a ignorar o último qubit do registrador, e consideraremos apenas

$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle.$$

Aplicamos a transformação de Hadamard a cada um dos $n$ primeiros qubits de $|\phi_2\rangle$ para obter

$$|\phi_3\rangle := H_n|\phi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} H_n|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} H_n|x\rangle.$$

Não é difícil ver que, para qualquer $x \in \{0,1\}^n$, temos

$$H_n|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y} |y\rangle, \tag{2.3}$$

onde $x \cdot y = \bigoplus_{j=0}^{n-1} x_j y_j$ e $\oplus$ é a operação ou-exclusivo.

Desta forma, temos

$$\begin{aligned}
|\phi_3\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \left[ \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y} |y\rangle \right] \\
&= \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{f(x)\oplus(x\cdot y)} |y\rangle.
\end{aligned}$$

Observe que a amplitude $\alpha_0$ do vetor básico $|0\rangle$ é

$$\alpha_0 = \sum_{x\in\{0,1\}^n} \frac{(-1)^{f(x)}}{2^n}.$$

Assim, se $f$ é constante, então $\alpha_0 = (-1)^{f(0)}$ e portanto $|\phi_3\rangle = (-1)^{f(0)}|0\rangle$. Já se $f$ é balanceada, então $\alpha_0 = 0$. Podemos então medir os $n$ qubits de $|\phi_3\rangle$

para descobrir se $f$ é balanceada ou constante: se todos os qubits do registrador valerem $|0\rangle$, então $f$ é constante; caso contrário, $f$ é balanceada.

O algoritmo apresentado acima resolve o problema de Deutsch-Jozsa com apenas uma chamada à caixa preta de $f$. É claro que, para um computador determinístico resolver o mesmo problema exatamente, são necessárias no pior caso $2^{n-1}+1$ chamadas a $f$ e portanto esse problema não pode ser resolvido em tempo polinomial no modelo determinístico. Entretanto, no modelo probabilístico, existe um algoritmo, descrito a seguir, que resolve o problema em tempo polinomial se for permitido um erro unilateral arbitrariamente pequeno.

Considere um algoritmo probabilístico que inicialmente sorteia $n$ números inteiros $x_1, \ldots, x_n$ tais que $0 \leq x_i < 2^n$ para todo $i$. Depois são feitas $n$ chamadas à caixa preta: para cada $i$, fazemos $y_i = f(x_i)$. Finalmente, verificamos se $y_i = y_{i+1}$ para $i = 1, \ldots, n-1$. Se este for o caso, então o algoritmo responde que $f$ é constante. Caso contrário, o algoritmo responde que $f$ é balanceada.

Note que, se o algoritmo afirma que $f$ é balanceada, então $f$ com certeza é balanceada. Entretanto, pode ser que o algoritmo afirme que $f$ é constante mesmo que $f$ seja balanceada. Vamos medir a probabilidade de ocorrência desse evento, ou seja, de o algoritmo responder que $f$ é constante sendo que $f$ é balanceada.

Fixada uma função $f$ balanceada, temos uma bipartição $\{F_0, F_1\}$ de $\{0,1\}^n$ com $|F_0| = |F_1|$, como na definição de função balanceada acima. Para $i = 1, \ldots, n$, defina a variável aleatória $X_i$ com valor 0 se $x_i \in F_0$ e 1 se $x_i \in F_1$. É claro que $\mathbb{P}[X_i = 0] = \mathbb{P}[X_i = 1] = 1/2$ para todo $i$, já que $|F_0| = |F_1|$. A probabilidade de o algoritmo responder que $f$ é constante é

$$\begin{aligned} q &:= \mathbb{P}[X_1 = 0, X_2 = 0, \ldots, X_n = 0] + \mathbb{P}[X_1 = 1, X_2 = 1, \ldots, X_n = 1] \\ &= \prod_{i=1}^{n} \mathbb{P}[X_i = 0] + \prod_{i=1}^{n} \mathbb{P}[X_i = 1] = \frac{1}{2^n} + \frac{1}{2^n} = \frac{1}{2^{n-1}}, \end{aligned}$$

de modo que $q \leq 1/2$ se $n \geq 2$ e portanto o problema de decidir se $f$ é balanceada está na classe de complexidade **RP**, que será apresentada na parte II deste texto.

Vamos mostrar agora uma variante do problema de Deutsch-Jozsa proposta por Bernstein e Vazirani [BV93].

Dado $a \in \{0,1\}^n$, considere a função $f_a : \{0,1\}^n \longrightarrow \{0,1\}$ dada por $f_a(x) := x \cdot a$. Suponha que a função $f_a$ é dada como uma caixa preta. O problema consiste em determinar $a$.

Note que $f_a$ é constante se $a = 0^n$ e $f_a$ é balanceada caso contrário (mas não é verdade que, se $f$ é garantidamente constante ou balanceada, então $f$ é igual a $f_a$ para algum $a \in \{0,1\}^n$).

A solução dada por Cleve, Ekert, Macchiavello e Mosca [CEMM98] para essa variante do problema é quase idêntica ao algoritmo que resolve o problema de Deutsch-Jozsa. Em particular, o algoritmo faz uma única chamada à caixa preta $U_{f_a}$. Considere o registrador $|\phi_3\rangle$ obtido como naquele algoritmo:

$$\begin{aligned}|\phi_3\rangle &= \frac{1}{2^n}\sum_{x,y\in\{0,1\}^n}(-1)^{f(x)\oplus(x\cdot y)}|y\rangle = \frac{1}{2^n}\sum_{x,y\in\{0,1\}^n}(-1)^{(x\cdot a)\oplus(x\cdot y)}|y\rangle\\ &= \frac{1}{2^n}\sum_{x,y\in\{0,1\}^n}(-1)^{x\cdot(a\oplus y)}|y\rangle.\end{aligned}$$

Observe que a amplitude do estado básico $|a\rangle$ é

$$\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{x\cdot 0} = 1$$

e portanto $|\phi_3\rangle = |a\rangle$. Portanto, basta medir os $n$ qubits do registrador para se obter $a$.

## 2.5 O problema de Simon

Seja $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ uma função. Dizemos que $f$ é *dois-para-um* se existe um vetor $s \in \{0,1\}^n$, $s \neq 0^n$, tal que, para quaisquer $x, x' \in \{0,1\}^n$, $x \neq x'$, tivermos $f(x) = f(x')$ se, e somente se, $x' = x \oplus s$.

O problema de Simon, também conhecido como problema XOR de Simon, consiste no seguinte:

**Problema 2.7 (Simon)** Seja $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ uma função dada como uma caixa preta, com a garantia de que $f$ é bijetora ou dois-para-um. Determine se $f$ é bijetora ou dois-para-um. Caso $f$ seja dois-para-um, determine também o vetor $s$ como especificado na definição acima.

A solução proposta por Simon [Sim97] resolve o problema em tempo esperado polinomial. O algoritmo consiste essencialmente em algumas repetições do procedimento HADAMARD-TWICE, que descrevemos a seguir.

Seja $|\phi_0\rangle$ um registrador de $2n$ qubits inicializado com $\left|0^n, 0^n\right\rangle$. Aplique a transformação de Hadamard a cada um dos $n$ primeiros qubits de $|\phi_0\rangle$, para obter

$$|\phi_1\rangle := (H_n \otimes I_n)|\phi_0\rangle = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}\left|x, 0^n\right\rangle.$$

Aplicamos agora a caixa preta $U_f$, que leva $|x,y\rangle$ a $\big|x, y \oplus f(x)\big\rangle$, a $|\phi_1\rangle$:

$$|\phi_2\rangle := U_f|\phi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \big|x, f(x)\big\rangle.$$

Mais uma vez aplicamos a transformação de Hadamard a cada um dos $n$ primeiros qubits, obtendo, de acordo com (2.3) da página 38,

$$|\phi_3\rangle := (H_n \otimes I_n)|\phi_2\rangle = \frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot y}\big|y, f(x)\big\rangle.$$

Faça agora uma medição de $|\phi_3\rangle$ para obter um par $\big(y, f(x)\big)$. O procedimento HADAMARD-TWICE devolve então o vetor $y$ observado.

Antes de descrever o algoritmo de Simon, vamos calcular as amplitudes dos estados básicos no vetor $|\phi_3\rangle$ acima. Suponha que $f$ é bijetora. Então todos os possíveis estados $\big|y, f(x)\big\rangle$ da superposição $|\phi_3\rangle$ são distintos e têm a mesma amplitude, que é $1/2^n$.

Agora considere o caso em que $f$ é dois-para-um e seja $s$ o vetor tal que $f(x) = f(x') \Leftrightarrow x' = x \oplus s$. Então os estados $\big|y, f(x)\big\rangle$ e $\big|y, f(x \oplus s)\big\rangle$ são idênticos e, portanto, sua amplitude total é

$$\alpha(x,y) = \frac{1}{2^n}\Big((-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y}\Big).$$

Se $s \cdot y \equiv 0 \pmod 2$, então $(x \oplus s) \cdot y \equiv (x \cdot y) \oplus (s \cdot y) \pmod 2$ e, portanto, $(x \oplus s) \cdot y \equiv x \cdot y \pmod 2$, de modo que $\alpha(x,y) = \pm 1/2^{n-1}$. Caso contrário, é óbvio que $\alpha(x,y) = 0$. Assim, um vetor $y$ devolvido pelo procedimento HADAMARD-TWICE com certeza satisfaz $s \cdot y \equiv 0 \pmod 2$.

O algoritmo de Simon repete o procedimento HADAMARD-TWICE até obter vetores $y_1, \ldots, y_{n-1}$ linearmente independentes. Após a obtenção de vetores $y_1, \ldots, y_{n-1}$ linearmente independentes, o algoritmo resolve o sistema de equações $s \cdot y_i \equiv 0 \pmod 2$ para todo $1 \leq i \leq n-1$. O subespaço das soluções deste sistema tem dimensão 1 (consulte [MK01] para um estudo mais detalhado acerca de espaços vetoriais definidos sobre corpos finitos) e portanto contém um único vetor não-nulo. O algoritmo encontra tal vetor $s^*$ e, se $f(0^n) = f(s^*)$, devolve a resposta “dois-para-um”; senão, devolve “bijetora”.

Resta provarmos agora que o número esperado de repetições do procedimento HADAMARD-TWICE é polinomial em $n$. Vamos fazer a análise do caso em que $f$ é dois-para-um. A análise do caso em que $f$ é bijetora é completamente análoga.

Para cada $1 \leq i \leq n-1$, defina a variável aleatória $X_i$ da seguinte forma. Suponha que $y_1, \ldots, y_{i-1}$ são os vetores linearmente independentes obtidos até um dado instante. Então o valor de $X_i$ é o número de vezes que o procedimento HADAMARD-TWICE é chamado até que se obtenha um vetor $y_i$ que não seja combinação linear de $y_1, \ldots, y_{i-1}$. É claro então que a variável aleatória $X$ definida como $\sum_i X_i$ é justamente o número total de chamadas ao procedimento HADAMARD-TWICE até a obtenção de $n-1$ vetores linearmente independentes.

Fixe $i$. É claro que $X_i$ segue a distribuição geométrica com parâmetro $p$, onde $p$ é a probabilidade do vetor devolvido por HADAMARD-TWICE não ser combinação linear de $y_1, \ldots, y_{i-1}$. Já observamos que o espaço amostral dos vetores que podem ser devolvidos pelo procedimento é $S := \{y \in \mathbb{Z}_2^n : s \cdot y \equiv 0 \pmod 2\}$ e que a distribuição de probabilidade nesse espaço é uniforme. Como $s \neq 0^n$ e $S$ é justamente o subespaço das soluções do sistema $s \cdot y \equiv 0 \pmod 2$, então $S$ tem dimensão $n-1$ e, portanto, $2^{n-1}$ elementos. É claro também que o conjunto de todos os vetores de $S$ que são combinação linear de $y_1, \ldots, y_{i-1}$ tem cardinalidade $2^{i-1}$ (note que $\{y_1, \ldots, y_{i-1}\} \subseteq S$). Desta forma, temos que

$$p := \frac{2^{n-1} - 2^{i-1}}{2^{n-1}} = \frac{2^{i-1}(2^{n-i} - 1)}{2^{n-1}} = 2^{i-n}(2^{n-i} - 1).$$

Mas então

$$\mathbb{E}\left[X_i\right] = 1/p = \frac{2^{n-i}}{2^{n-i} - 1} \leq 2,$$

pois $1 \leq i \leq n-1$.

É imediato que

$$\mathbb{E}\left[X\right] = \mathbb{E}\Big[\sum_{i=1}^{n-1} X_i\Big] = \sum_{i=1}^{n-1} \mathbb{E}\left[X_i\right] = O(n).$$

Portanto o tempo esperado de execução do algoritmo de Simon é $O\big(nT_f(n) + nG(n)\big)$, onde $T_f(n)$ é o tempo de execução da caixa preta $U_f$ e $G(n)$ é o tempo necessário para se resolver um sistema linear com $n$ incógnitas e $n-1$ equações, sobre o espaço vetorial $\mathbb{Z}_2^n$. Observe que estamos contabilizando o tempo para verificar se um conjunto é ou não linearmente independente através da função que resolve sistemas lineares.

É evidente que o algoritmo pode nunca terminar. Mas é muito fácil convertê-lo em um algoritmo que sempre termina, mas que dá a resposta com uma probabilidade limitada de erro. Basta fixar o número de chamadas ao procedimento HADAMARD-TWICE e, caso não se obtenha $n-1$ vetores linearmente independentes com este número de chamadas, dar uma resposta qualquer.

Brassard e Høyer [BH97] apresentaram um algoritmo que sempre produz a resposta certa para o problema de Simon e cujo tempo de execução é polinomial no pior caso.

# Capítulo 3

# O algoritmo de fatoração de Shor

O algoritmo de Shor resolve o problema da fatoração de inteiros em primos e consome tempo polinomial no tamanho da entrada. Apresentamos a seguir os detalhes fundamentais do funcionamento deste algoritmo.

## 3.1 Visão geral do algoritmo

O algoritmo de Shor [Sho97] é um algoritmo quântico que, dado um inteiro $n$ composto ímpar que não é potência de primo, devolve um fator de $n$ com probabilidade limitada de erro.

As restrições para o valor de $n$ não representam problema algum. De fato, é trivial encontrar um fator de um número par. Além disso, é fácil desenvolver um algoritmo eficiente que decide se $n = a^k$, para inteiros $a$ e $k > 1$, e que devolve $a$ e $k$ neste caso. Podemos, por exemplo, utilizar a idéia ingênua de, para cada $k$ desde $\lg n$ até 2, usar busca binária para determinar se existe um inteiro $r$ tal que $r^k = n$. Uma análise superficial mostra que o consumo de tempo deste algoritmo é $O\big((\lg n)^6\big)$, que é polinomial em $\lg n$. Existem algoritmos muito melhores para resolver este problema, como o apresentado por Bernstein [Ber98].

Ademais, podemos verificar em tempo polinomial se $n$ é composto, utilizando o algoritmo AKS. Outra opção é executar testes de primalidade probabilísticos um número suficiente de vezes. Na prática, isso é mais eficiente, pois os testes probabilísticos são, em geral, mais simples e rápidos que o AKS. O teste de Miller-Rabin, apresentado na seção D.2, é uma ótima escolha para esta verificação, por ser de fácil implementação e ter complexidade de tempo $O\big(\lg^3 n\big)$, com uma constante pequena escondida pela notação assintótica.

Como $n$ é produto de no máximo $\lg n$ inteiros, o algoritmo de Shor pode ser utilizado para resolver o problema da fatoração em tempo polinomial no tamanho da entrada.

O algoritmo de Shor baseia-se numa redução do problema da busca de um fator de $n$ ao problema da busca do período de uma seqüência. Como a redução utiliza aleatorização, é possível que ela falhe, isto é, que nenhum fator de $n$ seja encontrado. Porém, a probabilidade de ocorrência deste evento é limitada. Na seção 3.2 apresentamos essa redução e limitamos a probabilidade de falha.

Na seção 3.3, apresentamos um algoritmo quântico eficiente para a busca do período da seqüência gerada pela redução. Esse algoritmo utiliza a transformada quântica de Fourier, que pode ser implementada eficientemente, como mostramos na seção 3.4.

O algoritmo de busca de período apresentado na seção 3.3 pode ser facilmente generalizado para buscar eficientemente o período de qualquer seqüência. Mais formalmente, dado um oráculo $U_f$ que computa uma função $f$ de $\{0, \ldots, 2^m - 1\}$ em $\{0, \ldots, 2^a - 1\}$ com período $r$, a generalização do algoritmo faz uma única chamada a $U_f$ e usa um circuito quântico de tamanho polinomial em $m$ para descobrir $r$ com probabilidade limitada de erro.

## 3.2 Redução à busca do período

Seja $n$ um inteiro composto ímpar que não é potência de primo. Vamos mostrar como reduzir o problema de encontrar um fator de $n$ ao problema de encontrar o período de uma função. Essa redução utiliza aleatorização, de modo que precisaremos limitar a probabilidade de falha do procedimento.

**Algoritmo** Shor $(n)$

1 escolha um inteiro $1 < x < n$ aleatoriamente
2 se $\text{mdc}(x, n) > 1$
3 então devolva $\text{mdc}(x, n)$
4 seja $r$ o período da função $f(a) = x^a \bmod n$
5 se $r$ for ímpar ou $x^{r/2} \equiv -1 \pmod{n}$
6 então o procedimento falhou
7 devolva $\text{mdc}(x^{r/2} + 1, n)$

O algoritmo de Shor utiliza um único passo quântico: o cálculo do período da função na linha 4. Os demais passos podem ser efetuados em tempo polinomial no modelo tradicional, e portanto também no modelo quântico.

Agora vamos mostrar que, se a redução devolve uma resposta, ela está correta. Depois vamos delimitar superiormente a probabilidade de falha deste procedimento, isto é, a probabilidade de o algoritmo terminar na linha 6.

Começamos observando que, se o algoritmo executa a linha 3, então o valor devolvido de fato é um fator de $n$.

Já se $\mathrm{mdc}(x, n) = 1$, então $x$ está em $\mathbb{Z}_n^*$, o grupo multiplicativo módulo $n$, de modo que o corolário C.22 nos garante que a função $f(a) = x^a \bmod n$ é periódica com período dado pela ordem de $x$, módulo $n$. Isto é, o período $r$ é o tamanho do subgrupo de $\mathbb{Z}_n^*$ gerado por $x$. Pelo teorema C.21,

$$r \text{ é o menor inteiro positivo tal que } x^r \equiv 1 \pmod{n}. \tag{3.1}$$

Observe que, se $r$ é par e $x^{r/2} \not\equiv -1 \pmod{n}$, então $x^{r/2}$ é uma raiz quadrada não-trivial de 1, módulo $n$. Não precisamos verificar se $x^{r/2} \equiv 1 \pmod{n}$, pois $r$ é o menor inteiro positivo tal que $x^r \equiv 1 \pmod{n}$, como já foi notado. Aplicando o teorema C.38, vemos que $\mathrm{mdc}(x^{r/2}+1, n)$ e $\mathrm{mdc}(x^{r/2}-1, n)$ são ambos fatores de $n$. Assim, o valor devolvido na linha 7 é, de fato, um fator de $n$.

Resta limitarmos a probabilidade de falha do procedimento, ou seja, dado um inteiro $1 < x < n$ escolhido aleatoriamente com probabilidade uniforme, precisamos limitar a probabilidade de que $r$, a ordem de $x$, módulo $n$, seja ímpar ou satisfaça $x^{r/2} \equiv -1 \pmod{n}$ se for par.

Suponha que a fatoração de $n$ em primos é dada por $n = \prod_{i=1}^{m} p_i^{k_i}$, onde $p_i$ é primo e $k_i \geq 1$ para todo $i$ e $m > 1$, já que $n$ não é potência de primo. Para cada $i$, seja $n_i := {p_i}^{k_i}$, de modo que $n = n_1 \cdots n_m$. Sejam $1 < x < n$ um inteiro,

$$r \text{ a ordem de } x, \text{ módulo } n,$$

e

$$r_i \text{ a ordem de } x, \text{ módulo } n_i,$$

para $i = 1, \ldots, m$. Pelo corolário C.31 ao teorema do resto chinês, a equação $x^r \equiv 1 \pmod{n}$ é equivalente ao sistema

$$\begin{array}{c} x^r \equiv 1 \pmod{n_1} \\ x^r \equiv 1 \pmod{n_2} \\ \vdots \\ x^r \equiv 1 \pmod{n_m}. \end{array} \tag{3.2}$$

Conforme o teorema C.21, a ordem $r_i$ de $x$, módulo $n_i$, é o menor inteiro positivo tal que $x^{r_i} \equiv 1 \pmod{n_i}$. Pelo corolário C.22, temos $x^r \equiv x^{r_i} \pmod{n_i}$ se, e somente se, $r \equiv r_i \pmod{r_i}$. Então $r$ é múltiplo de $r_i$ para todo $i$. Segue de (3.1) que

$$r = \mathrm{mmc}(r_1, \ldots, r_m). \tag{3.3}$$

Sejam $c_i$ e $q_i$ tais que

$$r_i = 2^{c_i} q_i, \text{ com } q_i \text{ ímpar}, \tag{3.4}$$

para $i = 1, \ldots, m$. É fácil ver que $r$ é ímpar se, e somente se, $r_i$ é ímpar para todo $i$, isto é, se, e somente se, $c_i = 0$ para todo $i$.

Suponha agora que $r_i$ é par para algum $i$. Então $r$ é par. Vamos descobrir em que condições temos $x^{r/2} \equiv -1 \pmod{n}$. Novamente, pelo corolário C.31 ao teorema do resto chinês, a equação $x^{r/2} \equiv -1 \pmod{n}$ é equivalente ao sistema

$$\begin{aligned} x^{r/2} &\equiv -1 \pmod{n_1} \\ x^{r/2} &\equiv -1 \pmod{n_2} \\ &\vdots \\ x^{r/2} &\equiv -1 \pmod{n_m}. \end{aligned} \tag{3.5}$$

Suponha que existam $i, j \in \{1, \ldots, m\}$ tais que $c_i > c_j$. Então $r = 2r_j u$ para algum inteiro $u$, pela equação (3.3). Segue que $x^{r/2} \equiv x^{r_j u} \pmod{n_j}$. Mas então temos $x^{r/2} \equiv 1 \pmod{n_j}$, de modo que $x^{r/2} \not\equiv -1 \pmod{n}$. Portanto, para que $r$ seja par e $x^{r/2} \equiv -1 \pmod{n}$, é necessário que $c_1 = c_2 = \cdots = c_m > 0$.

Estabelecemos assim que, se o procedimento falha, então $c_1 = \cdots = c_m$. Vamos limitar a probabilidade de ocorrência desse evento, dada a escolha aleatória de $x$.

Pelo teorema C.29 chinês do resto, existe uma bijeção entre $\mathbb{Z}_n$ e o produto cartesiano $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_m}$:

$$\mathbb{Z}_n \ni x \longleftrightarrow (x_1, \ldots, x_m) \in \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_m}. \tag{3.6}$$

Assim, escolher um inteiro $0 \leq x < n$ aleatoriamente é equivalente a escolher, independentemente para cada $1 \leq i \leq m$, um inteiro $0 \leq x_i < n_i$. Vamos supor que $\mathrm{mdc}(x, n) = 1$, já que a redução não se aplica se $\mathrm{mdc}(x, n) > 1$. Então $x \in \mathbb{Z}_n^*$. É fácil ver que $x \in \mathbb{Z}_n^*$ se, e somente se, $x_i \in \mathbb{Z}_{n_i}^*$ para todo $i$. Portanto estamos escolhendo aleatoriamente um $x_i \in \mathbb{Z}_{n_i}^*$, independentemente, para cada $i = 1, \ldots, m$:

$$\mathbb{Z}_n^* \ni x \longleftrightarrow (x_1, \ldots, x_m) \in \mathbb{Z}_{n_1}^* \times \cdots \times \mathbb{Z}_{n_m}^*. \tag{3.7}$$

Para cada $i = 1, \ldots, m$, o grupo $\mathbb{Z}_{n_i}^*$ é cíclico, conforme o teorema C.28, pois $n_i = {p_i}^{k_i}$ com $p_i$ primo. Seja $g_i$ um gerador de $\mathbb{Z}_{n_i}^*$, para cada $i$. Temos que

$$x_i = g_i^{l_i}, \text{ com } 0 \leq l_i < \phi(n_i), \text{ para } i = 1, \ldots, m. \tag{3.8}$$

Estamos então escolhendo aleatoriamente um $0 \leq l_i < \phi(n_i)$ para cada $i$, independente e uniformemente:

$$\mathbb{Z}_n^* \ni x \longleftrightarrow (g_1^{l_1}, \ldots, g_m^{l_m}) \in \mathbb{Z}_{n_1}^* \times \cdots \times \mathbb{Z}_{n_m}^*, \tag{3.9}$$

Para $i = 1, \ldots, m$, sejam $d_i$ e $s_i$ tais que

$$\phi(n_i) = 2^{d_i} s_i, \text{ com } s_i \text{ ímpar},$$

e lembre-se que

$$r_i = 2^{c_i} q_i \text{ é a ordem de } x_i, \text{ módulo } n_i, \text{ com } q_i \text{ ímpar.}$$

Note que $\phi(n_i) = \phi\big(p_i^{k_i}\big) = p_i^{k_i - 1}(p_i - 1)$, conforme o teorema C.32, e portanto $d_i > 0$, pois $p_i$ é ímpar.

Pelo teorema C.17 de Lagrange, $r_i$ divide $\phi(n_i)$, e portanto $c_i \leq d_i$. Pelo teorema C.21, $r_i$ é o menor inteiro positivo tal que $g_i^{l_i r_i} \equiv 1 \pmod{n_i}$. Segue do corolário C.22 que $l_i r_i = 2^{c_i} q_i l_i$ é múltiplo de $\phi(n_i) = 2^{d_i} s_i$. Se $l_i$ é ímpar, então devemos ter $c_i = d_i$, pois $q_i l_i$ é ímpar. Já se $l_i$ é par, então necessariamente teremos $c_i < d_i$: se $c_i = d_i$, então $r_i/2$ também é inteiro e $l_i r_i/2$ também é múltiplo de $\phi(n_i)$, um absurdo. Portanto, a probabilidade de que $c_i = c$, para qualquer $c$, é limitada por $1/2$, já que $0 \leq l_i < \phi(n_i)$ e $\phi(n_i)$ é par.

Concluímos então que a probabilidade de falha dessa redução, ou, na verdade, a probabilidade de que $c_1 = \cdots = c_m$ é, no máximo, $1 - 1/2^{m-1} \leq 1/2$, pois $m > 1$ e a escolha de cada $c_i$ é independente de todas as outras.

## 3.3 Busca do período

Seja $n$ um inteiro composto ímpar que não é potência de primo e seja $1 < x < n$ um inteiro relativamente primo a $n$. Vamos apresentar um algoritmo quântico que descobre, com probabilidade limitada de erro, a ordem de $x$, módulo $n$, que, de acordo com o teorema C.21 e o corolário C.22, é o período da seqüência

$$\langle x^0 \bmod n, x^1 \bmod n, x^2 \bmod n, \ldots \rangle. \tag{3.10}$$

Seja $\beta := \lfloor \lg n \rfloor + 1$ o número de bits da representação binária de $n$ e seja $q = 2^l$ a única potência de 2 tal que $n^2 \leq q < 2n^2$. Vamos precisar de um registrador $|\phi\rangle$ com $l + \beta$ qubits. Os $l$ primeiros qubits formam o primeiro sub-registrador. Os $\beta$ qubits restantes farão parte do segundo sub-registrador. Todos os qubits do registrador $|\phi\rangle$ devem ser inicializados com $|0\rangle$.

Após a aplicação da transformação de Hadamard a cada um dos qubits do primeiro sub-registrador, o estado do registrador $|\phi\rangle$ será

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a,0\rangle. \tag{3.11}$$

Seja $U_f$ a transformação unitária que, para todo $0 \le a < q$, leva um estado básico $|a,0\rangle$, ao estado $|a, x^a \bmod n\rangle$. No apêndice C, mostramos um algoritmo para exponenciação modular que consome $O(\beta^3)$ operações sobre bits. Então, para todo $n$, existe um circuito com $O(\beta^3)$ portas, cada uma operando sobre no máximo um número fixo de qubits, que efetua a transformação $U_f$. Em outras palavras, $U_f$ pode ser implementada eficientemente (veja o artigo de Shor [Sho97] para mais detalhes).

Aplicando a transformação $U_f$ ao registrador $|\phi\rangle$, obtemos o estado

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a, x^a \bmod n\rangle. \tag{3.12}$$

Medindo o estado do segundo sub-registrador, obtemos algum $x^b \bmod n$, onde $0 \le b < q$. Com isso, o estado do registrador $|\phi\rangle$ colapsa para uma superposição dos estados básicos de (3.12) cujos $\beta$ qubits menos significativos representam $x^b \bmod n$.

Seja $r$ a ordem de $x$, módulo $n$. Então os valores de $0 \le a < q$ tais que $x^a \equiv x^b \pmod{n}$ são da forma $a_0 + jr$, com $0 \le a_0 < r$, já que, pelo corolário C.22, a seqüência (3.10) é periódica com período $r$. A medição do segundo sub-registrador em (3.12) selecionará os seguintes valores de $a$, no primeiro sub-registrador: $a_0, a_0+r, a_0+2r, \ldots, a_0+(A-1)r$, onde $A = \lceil q/r \rceil$. O estado do registrador $|\phi\rangle$ será então

$$\frac{1}{\sqrt{A}}\sum_{j=0}^{A-1}|a_0 + jr, x^b \bmod n\rangle.$$

De agora em diante, vamos ignorar o segundo sub-registrador. Então o estado de $|\phi\rangle$ será

$$\frac{1}{\sqrt{A}}\sum_{j=0}^{A-1}|a_0 + jr\rangle. \tag{3.13}$$

Note que os estados básicos de $|\phi\rangle$ que podem ser obtidos numa medição estão uniformemente espaçados a partir de $a_0$, com espaço $r$.

Seja $M_q$ a transformação unitária dada por

$$M_q : |x\rangle \longrightarrow \frac{1}{\sqrt{q}} \sum_{y=0}^{q-1} \exp(2\pi i x y / q) |y\rangle. \tag{3.14}$$

Vamos mostrar, na seção 3.4, que esta transformação, conhecida como matriz de Vandermonde, pode ser implementada eficientemente por um circuito quântico. A aplicação de $M_q$ ao registrador $|\phi\rangle$, dado por (3.13), gera o estado

$$\begin{aligned} |\phi\rangle &= \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} \left[ \frac{1}{\sqrt{q}} \sum_{y=0}^{q-1} \exp\left\{2\pi i (a_0 + jr) y / q\right\} |y\rangle \right] \\ &= \frac{1}{\sqrt{qA}} \sum_{y=0}^{q-1} \left[ \exp\{2\pi i a_0 y / q\} \sum_{j=0}^{A-1} \exp\{2\pi i j r y / q\} |y\rangle \right]. \end{aligned}$$

Então a amplitude de um estado básico $|y\rangle$ é

$$\frac{1}{\sqrt{qA}} \exp\{2\pi i a_0 y / q\} \sum_{j=0}^{A-1} \exp\{2\pi i j r y / q\}, \tag{3.15}$$

de modo que a probabilidade de obtenção de $|y\rangle$ numa medição de $|\phi\rangle$ é

$$p_y = \frac{1}{qA} \left| \sum_{j=0}^{A-1} \exp\{2\pi i j r y / q\} \right|^2. \tag{3.16}$$

Aqui vamos apenas analisar o caso em que $r$ é uma potência de 2, de modo que $A = q/r$. Os demais casos seguem a mesma idéia porém são mais técnicos.

Suponha que $y$ não é múltiplo de $A$. Então $ry/q$ não é inteiro, de modo que $\exp\{2\pi i r y / q\} \neq 1$. Pela fórmula da soma de uma progressão geométrica, temos

$$\begin{aligned} \sum_{j=0}^{A-1} \exp\{2\pi i j r y / q\} &= \sum_{j=0}^{A-1} \left( \exp\{2\pi i r y / q\} \right)^j \\ &= \frac{\left( \exp\{2\pi i r y / q\} \right)^A - 1}{\exp\{2\pi i r y / q\} - 1} \\ &= \frac{\exp\{2\pi i y\} - 1}{\exp\{2\pi i r y / q\} - 1} \quad = \quad 0, \end{aligned}$$

pois $A = q/r$.

Suponha agora que $y$ é um múltiplo de $A$. Então $ry/q$ é inteiro, de modo que $\exp\{2\pi ijry/q\} = 1$ para todo $0 \leq j < A$. Então

$$\sum_{j=0}^{A-1} \exp\{2\pi ijry/q\} = A.$$

Concluímos que a probabilidade de obtenção de $|y\rangle$ na medição do registrador $|\phi\rangle$ no estado (3.15) é

$$p_y = \begin{cases} 1/r, & \text{se } y \text{ é múltiplo de } q/r \\ 0, & \text{caso contrário.} \end{cases} \tag{3.17}$$

Assim, uma medição de $|\phi\rangle$ nos fornece um $y = cq/r$, com $0 \leq c < r$ escolhido equiprovavelmente. Teremos então um valor de $y$ satisfazendo $y/q = c/r$, onde $c$ e $q$ são conhecidos. Se $\text{mdc}(c, r) = 1$, então basta obter a fração irredutível correspondente a $y/q$ para chegarmos ao período $r$. A probabilidade de obtenção de um $0 \leq c < r$ com $\text{mdc}(c, r) = 1$ é $\phi(r)/r$. Pode-se provar [HW54] que existe uma constante $\delta$ tal que $\phi(r)/r > \delta/\log\log r$. Assim, a probabilidade de falha deste procedimento é, no máximo, $1 - \delta/\log\log r$.

Se repetirmos o procedimento acima $z := \log\log r/\delta$ vezes, a probabilidade de falha passará a ser $(1 - 1/z)^z \leq 1/e$, de modo que a probabilidade de sucesso é, no mínimo, $1 - 1/e$, uma constante. Portanto, obtemos o período $r$ com probabilidade limitada inferiormente por uma constante.

## 3.4 A transformada quântica de Fourier

Vamos ver agora que a transformação unitária $M_q$, dada por (3.14), pode ser implementada eficientemente sempre que $q$ for uma potência de 2. Ou seja, para todo $q = 2^m$, vamos mostrar que existe um circuito de tamanho polinomial em $m$, utilizando apenas portas quânticas que operam sobre um número fixo de qubits, cuja aplicação a um registrador com $m$ qubits é equivalente à aplicação da matriz $M_q$.

Primeiro vamos escrever o estado

$$\frac{1}{\sqrt{q}} \sum_{y=0}^{q-1} \exp(2\pi ixy/q)|y\rangle \tag{3.18}$$

de uma forma mais conveniente. Considere $q = 2^m$, com $m$ um inteiro positivo. Denotaremos por $(x_{m-1} \cdots x_0)_2$ a representação binária de $0 \leq x < 2^m$, ou seja, $x = \sum_{j=0}^{m-1} x_j 2^j$, com $x_j \in \{0, 1\}$ para todo $j$. Além disso, utilize $(0.\, x_1 \cdots x_p)_2$

para denotar a representação binária de $0 \leq x < 1$, isto é, $x = \sum_{j=1}^{p} x_j 2^{-j}$, com $x_j \in \{0, 1\}$ para todo $j$.

Então o estado (3.18) não é emaranhado e pode ser fatorado como

$$\begin{aligned}
&\left[\frac{1}{\sqrt{2}}\Big(|0\rangle + \exp\big\{2\pi i (0.\, x_0)_2\big\}|1\rangle\Big)\right] \otimes \\
&\qquad \left[\frac{1}{\sqrt{2}}\Big(|0\rangle + \exp\big\{2\pi i (0.\, x_1 x_0)_2\big\}|1\rangle\Big)\right] \otimes \cdots \otimes \\
&\qquad\qquad \left[\frac{1}{\sqrt{2}}\Big(|0\rangle + \exp\big\{2\pi i (0.\, x_{m-1} \cdots x_0)_2\big\}|1\rangle\Big)\right].
\end{aligned} \tag{3.19}$$

Para mostrar que o estado quântico (3.18) é o mesmo que o estado (3.19), vamos mostrar que, para todo estado básico $|y_{m-1} \cdots y_0\rangle$, temos

$$\begin{aligned}
&\exp\{2\pi i x y / 2^m\} |y_{m-1} \cdots y_0\rangle = \\
&\quad \Big(\exp\big\{2\pi i (0.\, x_0)_2 y_{m-1}\big\}|y_{m-1}\rangle\Big) \otimes \\
&\quad \Big(\exp\big\{2\pi i (0.\, x_1 x_0)_2 y_{m-2}\big\}|y_{m-2}\rangle\Big) \otimes \cdots \otimes \\
&\qquad \Big(\exp\big\{2\pi i (0.\, x_{m-1} \cdots x_0)_2 y_0\big\}|y_0\rangle\Big),
\end{aligned} \tag{3.20}$$

onde o lado direito da equação (3.20) mostra como se forma o estado básico $|y\rangle = |y_{m-1} \cdots y_0\rangle$ em (3.19). Será então suficiente mostrar que

$$\begin{aligned}
&\exp\{2\pi i x y / 2^m\} \\
&\quad = \exp\big\{2\pi i (0.\, x_0)_2 y_{m-1}\big\} \; \exp\big\{2\pi i (0.\, x_1 x_0)_2 y_{m-2}\big\} \cdots \\
&\qquad\quad \exp\big\{2\pi i (0.\, x_{m-1} \cdots x_0)_2 y_0\big\} \\
&\quad = \exp\Big\{2\pi i \big[(0.\, x_0)_2 y_{m-1} + (0.\, x_1 x_0)_2 y_{m-2} + \cdots + \\
&\qquad\qquad (0.\, x_{m-1} \cdots x_0)_2 y_0\big]\Big\}.
\end{aligned} \tag{3.21}$$

Observe que

$$\frac{yx}{2^m} = \frac{1}{2^m} \sum_{j=0}^{m-1} \left[ y_j 2^j \sum_{k=0}^{m-1} x_k 2^k \right] = \sum_{j=0}^{m-1} \left[ y_j \sum_{k=0}^{m-1} x_k 2^{k-m+j} \right]. \tag{3.22}$$

Na equação (3.21), o termo $xy/2^m$ aparece multiplicando $2\pi i$. Então apenas a parte fracionária de $xy/2^m$ é relevante: se $xy/2^m = u + r$, com $0 \leq r < 1$ e $u \in \mathbb{Z}$, então $\exp(2\pi i x y / 2^m) = \exp(2\pi i r)$. Na equação (3.22), para $k \geq m - j$, o valor $2^{k-m+j}$ é inteiro, de modo que podemos reescrever (3.22) como

$$\frac{yx}{2^m} = \sum_{j=0}^{m-1} \left[ y_j u_j + y_j \sum_{k=0}^{m-j-1} x_k 2^{k-m+j} \right], \tag{3.23}$$

onde $u_j$ é um inteiro. É fácil verificar agora que

$$\sum_{k=0}^{m-j-1} x_k 2^{k-m+j} = (0.\, x_{m-j-1} \cdots x_0)_2.$$

Mas então

$$xy/2^m = u + y_{m-1}(0.\, x_0)_2 + y_{m-2}(0.\, x_1 x_0)_2 + \cdots + y_0(0.\, x_{m-1} \cdots x_0)_2, \quad (3.24)$$

onde $u$ é um inteiro. A equação (3.21) segue imediatamente da equação (3.24), de modo que fica provado que o estado (3.18) pode ser fatorado como (3.19).

Considere o circuito quântico apresentado na figura 3.1, referente ao caso em que $m = 4$.

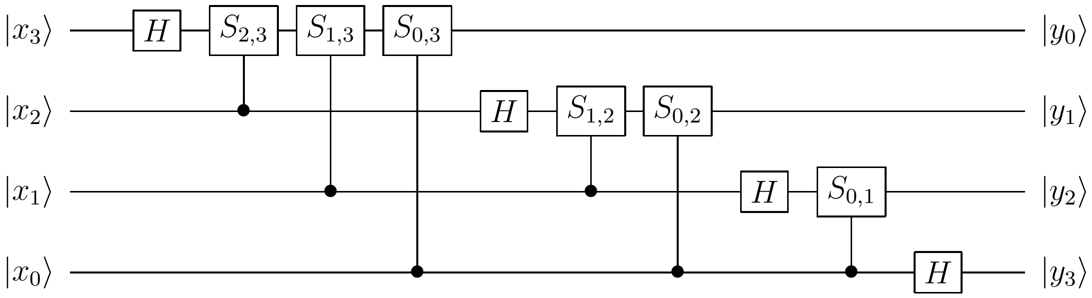


Figura 3.1: Circuito para a transformada quântica de Fourier, com $m = 4$.

Nesta figura, a matriz $S_{j,k}$, para $j < k$ é definida por

$$S_{j,k} = \begin{pmatrix} 1 & 0 \\ 0 & \exp\{2\pi i/2^{k-j+1}\} \end{pmatrix}. \quad (3.25)$$

É muito fácil ver como este circuito se estende para qualquer $m$. Para cada qubit $|x_k\rangle$, aplicamos a matriz de Hadamard, seguida de $k$ portas controladas por $S_{j,k}$, para todo $0 \le j < k$, onde a porta controlada por $S_{j,k}$ opera sobre os qubits $|x_k\rangle$ e $|x_j\rangle$.

Para ver que esse circuito de fato calcula a transformada quântica de Fourier, vamos analisar sua operação sobre o qubit $|x_3\rangle$ na figura 3.1. Após a aplicação da matriz de Hadamard, o estado deste qubit será $|0\rangle + \exp\big\{2\pi i(0.\, x_3)_2\big\}|1\rangle$. Note que estamos desprezando o fator de normalização $1/\sqrt{2}$, para maior clareza. Depois da aplicação da porta controlada por $S_{2,3}$, o estado passa a ser $|0\rangle + \exp\big\{2\pi i(0.\, x_3 x_2)_2\big\}|1\rangle$. Aplicando agora a porta controlada por $S_{1,3}$, teremos $|0\rangle + \exp\big\{2\pi i(0.\, x_3 x_2 x_1)_2\big\}|1\rangle$ e, por fim, após $S_{0,3}$ o estado será

$|0\rangle + \exp\left\{2\pi i(0.x_3 \cdots x_0)_2\right\}|1\rangle$. Mas isso é justamente $|y_0\rangle$, conforme a equação (3.19). Repetindo esses cálculos com os outros qubits, obteremos os estados desejados, de acordo com a equação (3.19).

Note que os qubits da saída deste circuito estão na ordem inversa. É óbvio que isso não representa qualquer problema para nós.

Observe também que o circuito utiliza $m(m+1)/2$ portas, o que é quadrático em $m$. Assim, a transformada quântica de Fourier pode ser implementada por um circuito de tamanho $O(m^2)$.

# Parte II

# Resultados de Complexidade

# Capítulo 4

# Máquinas de Turing

Agora mudamos de assunto, nos voltando à teoria de complexidade computacional. Os resultados dessa área são usualmente apresentados por meio do mais tradicional modelo de computação — a máquina de Turing (MT), que nada mais é que um computador bastante rudimentar com memória infinita.

Vamos apresentar três variantes desse modelo: a máquina de Turing determinística, a não-determinística e a probabilística. Depois disso, apresentamos a segunda formalização do modelo quântico de computação, a máquina de Turing quântica, fazendo um paralelo com o modelo clássico de computação. Como observamos na introdução, essa segunda formalização é equivalente à primeira. A demonstração desse fato não é trivial e não será mostrada nesse texto.

## 4.1 Máquina de Turing determinística

Uma máquina de Turing determinística (MTD) é composta por uma central de controle, uma cabeça de leitura e uma fita dividida em células. Essa fita tem um final à esquerda e contém infinitas células à direita.

Cada célula da fita armazena um símbolo pertencente a um conjunto finito $\Sigma$. O conjunto $\Sigma$ é chamado de *alfabeto* da máquina e contém, entre outros, dois símbolos especiais: o símbolo $\sqcup$, chamado de *branco*, e o símbolo $\triangleright$, que fica armazenado o tempo todo na célula mais à esquerda da fita.

A cabeça de leitura da máquina é um apontador móvel para uma determinada célula da fita. O conteúdo dessa célula pode ser lido e alterado pela máquina.

A cada instante, a central de controle da máquina está em um dos estados de um conjunto finito $Q$ de estados. No instante inicial, a máquina começa em um estado particular de $Q$, chamado de estado *inicial* e denotado por $s$. Existe ainda um conjunto especial de estados $H \subseteq Q$, chamados de estados *finais*.

Uma MTD funciona da seguinte maneira. Ela recebe como entrada uma cadeia de caracteres em $\left(\Sigma \setminus \{\sqcup, \rhd\}\right)^*$. Inicialmente a cabeça de leitura aponta para a célula mais à esquerda da fita, e a partir da célula seguinte está armazenada a entrada da máquina, seguida por brancos. A máquina efetua uma seqüência de passos até terminar a execução. Se $q$ é o estado corrente da máquina e $q$ está em $H$, então ela termina a execução. Do contrário, ela realiza três ações, determinadas pelo par $(q, \sigma)$, onde $\sigma$ é o símbolo de $\Sigma$ contido na célula que está sob a cabeça de leitura.

A primeira ação consiste na alteração do estado da máquina de $q$ para um estado $q'$ em $Q$. A segunda ação é a escrita de um símbolo $\sigma'$ na célula que está sob a cabeça de leitura. A terceira ação consiste no deslocamento da cabeça de leitura, que pode se mover uma posição para a esquerda, uma posição para a direita ou pode continuar apontando para a mesma célula.

A cabeça de leitura sempre desloca-se para a direita quando atinge a célula mais à esquerda e nunca altera o conteúdo dessa posição da fita. Por conveniência, assume-se que a máquina não pode escrever o símbolo $\rhd$ em qualquer posição da fita que não seja a célula mais à esquerda.

A cadeia de caracteres escrita na fita quando a máquina termina a execução, ignorando-se o símbolo $\rhd$ e os brancos à direita, é a saída da máquina para a entrada em questão.

Formalmente, define-se uma máquina de Turing determinística como uma quíntupla $M := (Q, \Sigma, \delta, s, H)$, onde $Q$ é o conjunto de estados, $\Sigma$ é o alfabeto de símbolos, $\delta$ é uma função de $(Q \setminus H) \times \Sigma$ em $Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$, $s \in Q$ é o estado inicial e $H \subseteq Q$ é o conjunto de estados finais. O conjunto $\{\leftarrow, \downarrow, \rightarrow\}$ descreve os possíveis movimentos da cabeça de leitura da máquina.

A função $\delta$, chamada usualmente de *função de transição*, descreve um passo arbitrário da máquina e satisfaz as restrições mencionadas acima. Suponha que a máquina esteja num estado $q$ em $Q \setminus H$ e que sob a cabeça de leitura esteja o símbolo $\sigma$. Se $\delta(q, \sigma) = (q', \sigma', d)$, o próximo estado será $q'$, o símbolo $\sigma'$ será escrito no lugar de $\sigma$ e a cabeça de leitura de $M$ moverá de acordo com $d$.

A *configuração atual* de $M$ é uma tripla $(w_1, q, w_2) \in \Sigma^* \times Q \times \Sigma^*$, onde $w_1$ é a palavra que aparece na fita à esquerda da cabeça de leitura, $q$ é o estado atual e $w_2$ é a palavra à direita da cabeça de leitura ignorando-se brancos à direita. A cabeça de leitura aponta para a posição que contém o último símbolo de $w_1$. A configuração inicial é a tripla $(\rhd, s, x)$, onde $x$ é a entrada para a máquina.

Dizemos que uma configuração $(w_1, q, w_2)$ *produz em $k$ passos* uma configuração $(w_1', q', w_2')$, denotado por $(w_1, q, w_2) \vdash_M^k (w_1', q', w_2')$, se a máquina sai da primeira configuração e vai para a segunda em exatos $k$ passos.

## 4.2 Máquinas de Turing não-determinísticas

A máquina de Turing não-determinística (MTND) é uma generalização da máquina de Turing determinística. Essa máquina tem um papel fundamental na teoria de complexidade pois está na base da definição da classe **NP**, conforme será mostrado posteriormente.

A maior parte das definições e idéias envolvidas na descrição das MTDs também se aplica às MTNDs. A diferença entre a MTND e a MTD está na função de transição e na maneira como as transições são feitas a cada passo. Nas máquinas determinísticas, existe uma única transição possível a partir de uma dupla $(q, \sigma)$, onde $q$ é um estado não-final e $\sigma$ é um símbolo em $\Sigma$. Já nas máquinas não-determinísticas, pode existir mais de uma transição válida a partir de uma dupla $(q, \sigma)$. Em cada passo da MTND, uma transição válida é escolhida arbitrariamente e é executada.

O número de transições válidas a partir de uma dupla $(q, \sigma)$ é limitado superiormente por $3 \times |\Sigma| \times |Q|$. Logo, o número de configurações que podem ser produzidas a partir de cada configuração também é limitado superiormente por $3 \times |\Sigma| \times |Q|$. A transição numa MTND é portanto uma *relação* e não necessariamente uma função.

Formalmente, uma máquina de Turing não-determinística é uma quíntupla $M := (Q, \Sigma, \Delta, s, H)$, onde $Q$ é o conjunto de estados, $\Sigma$ é o alfabeto de símbolos, $\Delta$ é uma relação de $(Q \setminus H) \times \Sigma$ em $Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$, $s \in Q$ é o estado inicial e $H \subseteq Q$ é o conjunto de estados finais.

É evidente que as MTDs são casos particulares de MTNDs em que $\Delta$ é uma função, já que toda função é uma relação.

Uma transição $(q', \sigma', d)$ em $\Delta(q, \sigma)$, onde $q \in Q$ e $\sigma \in \Sigma$, é interpretada como antes. Assim, $q'$ será o próximo estado da máquina, $\sigma'$ deve ser escrito no lugar de $\sigma$ e $d$ indicará o deslocamento da cabeça de leitura. Tal transição será válida somente se $(q', \sigma', d) \in \Delta(q, \sigma)$. A definição de configuração é igual à que usamos na definição da máquina de Turing determinística. Para MTNDs, dizemos que uma configuração $(w_1, q, w_2)$ produz a configuração $(w_1', q', w_2')$ em $k$ passos se, estando a máquina na primeira configuração, após $k$ passos ela pode estar na segunda a partir de uma seqüência de transições válidas.

A existência de várias transições possíveis para cada par $(q, \sigma)$ numa MTND permite a ocorrência de computações distintas para uma mesma entrada. Como as MTNDs podem ter várias computações válidas possíveis para uma mesma entrada, elas podem produzir saídas diferentes para uma mesma entrada. Comentaremos mais sobre isso quando falarmos das linguagens decididas por uma máquina de Turing.

## 4.3 Máquina de Turing probabilística

Uma máquina de Turing probabilística (MTP) é uma MTND que funciona de maneira diferente. A cada passo, em vez de uma transição ser escolhida arbitrariamente dentre todas as transições aplicáveis, escolhe-se uma com probabilidade uniforme. Assim, pode-se falar na probabilidade da MTP produzir, numa computação para uma certa entrada, uma saída específica. Todas as definições dadas para as MTNDs aplicam-se de maneira natural a uma MTP. Observe que uma MTD é uma MTP em que, a cada passo, há apenas uma transição aplicável.

Uma MTP corresponde à formalização do conceito de um computador acoplado a um gerador de símbolos aleatórios.

É possível descrever o funcionamento de uma MTP postergando as escolhas aleatórias para o final. Ou seja, pode-se calcular a probabilidade de se estar em uma configuração específica a cada passo e só ao final fazer um único sorteio para determinar a configuração em que a máquina termina.

## 4.4 Máquina de Turing quântica

A definição de uma máquina de Turing quântica (MTQ) assemelha-se à de uma MTP. Uma máquina de Turing quântica é uma MTND $M = (Q, \Sigma, \Delta, s, H)$ com a relação $\Delta$ substituída por uma função

$$\alpha : Q \times \Sigma \times Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\} \longrightarrow \mathbb{C}$$

tal que, para cada $(q, \sigma)$ em $Q \times \Sigma$, vale que

$$\sum_{(q',\sigma',d)\in T} |\alpha(q, \sigma, q', \sigma', d)|^2 = 1,$$

onde $T := Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$.

Cada número $\alpha(q, \sigma, q', \sigma', d)$ é chamado de *amplitude*. Note que $\alpha$ determina uma distribuição de probabilidade nas possíveis transições aplicáveis a um par $(q, \sigma)$.

Define-se *superposição de configurações* como uma combinação linear de configurações, onde o coeficiente de uma configuração $c$ é um número complexo $\alpha_c$ e $\sum_c |\alpha_c|^2 = 1$, sendo que o somatório é sobre todas as configurações $c$ que estão na superposição. O coeficiente $\alpha_c$ é a *amplitude* da configuração $c$.

A primeira diferença no funcionamento de uma MTQ frente às máquinas anteriores é que esta, a cada instante, encontra-se numa superposição de configurações. Para que a transição de uma MTQ esteja bem definida, é preciso que a máquina satisfaça a seguinte propriedade: se, numa superposição obtida depois de $k$ passos a partir da configuração inicial, há uma configuração com amplitude não-nula num estado final, então todas as configurações com amplitude não-nula nessa superposição estão num estado final.

Se todas as configurações da MTQ com amplitude não-nula forem finais, a MTQ termina a execução. Caso contrário, ela efetua uma transição, que consiste no seguinte. Digamos que $\psi := \sum_{j=1}^{t} \alpha_j c_j$ é a superposição corrente, onde $\sum_{j=1}^{t} |\alpha_j|^2 = 1$ e $\alpha_j \neq 0$ para todo $j$. Para cada $j$, seja $C_j$ o conjunto das configurações que podem ser obtidas de $c_j$ em um passo por meio de uma transição cuja amplitude é não-nula. Para cada $c$ em $C_j$, seja $\alpha_c^j$ a amplitude correspondente à transição de $c_j$ para $c$. Então, temos que a superposição resultante da transição é $\psi' = \sum_{j=1}^{t} \alpha_j \sum_{c \in C_j} \alpha_c^j c$.

Quando uma mesma configuração $c$ aparece em mais de um conjunto $C_j$ e $\left|\sum_{j:c \in C_j} \alpha_j \alpha_c^j\right|^2 \neq \sum_{j:c \in C_j} |\alpha_j \alpha_c^j|^2$, dizemos que houve *interferência*. A interferência é *negativa* se o lado esquerdo é menor que o direito e *positiva* caso contrário. O fenômeno de interferência é o principal responsável pela diferença entre uma MTQ e uma MTP.

A segunda diferença no funcionamento de uma MTQ frente às máquinas anteriores é que, ao final de sua execução, a MTQ efetua o que chamamos de *medição*. Digamos que $\psi := \sum_{j=1}^{t} \alpha_j c_j$ é a superposição final da máquina. Uma medição consiste na escolha aleatória de uma configuração $c = c_j$ com probabilidade $|\alpha_j|^2$, e na transição da superposição $\psi$ para a superposição $\psi' := c$, ou seja, para a superposição em que apenas a configuração $c$ tem amplitude não-nula (mais exatamente, $c$ tem amplitude 1 em $\psi'$). A saída da MTQ é o que está escrito na fita após a medição. Assim como uma MTP, uma MTQ pode produzir diferentes saídas para uma mesma entrada, cada uma com uma probabilidade.

Chamamos de *paralelismo quântico* à capacidade da MTQ estar numa superposição de configurações, e, num passo, efetuar transições múltiplas, envolvendo diversas configurações. Potencialmente é possível explorar o fenômeno de interferência bem como o paralelismo quântico para se obter algoritmos quânticos eficientes para problemas considerados difíceis no modelo clássico de computação.

## 4.5 Recursos e linguagens

Apresentamos algumas definições do modelo clássico de computação com o intuito de estabelecer um paralelo durante a apresentação dos correspondentes quânticos desses resultados e definições.

Para isso, definiremos o significado de tempo e espaço consumido nas máquinas do modelo clássico. Em seguida, falaremos das linguagens decididas por cada uma dessas máquinas.

## Tempo consumido pelas máquinas de Turing

Durante a descrição da MTD, foi definido que $(w_1, q, w_2) \vdash_M^k (w_1', q', w_2')$ se a máquina $M$ sai da primeira configuração e vai para a segunda em exatos $k$ passos. Se $(w_1, q, w_2)$ é a configuração inicial de $M$ e $q'$ é um de seus estados finais, então dizemos que $k$ é o tempo consumido por $M$ para a entrada $w_2$.

Dada uma MTD M, se existe um polinômio $p(n) : \mathbb{N} \longrightarrow \mathbb{N}$ tal que, para qualquer entrada $x$, o tempo consumido por $M$ é limitado superiormente por $p(|x|)$, onde $|x|$ denota o comprimento da palavra $x$, então dizemos que $M$ é *polinomialmente limitada*.

Se $M$ é uma MTND, o maior tempo consumido em uma computação válida de $M$ para uma entrada determinada é considerado o tempo consumido por $M$ para essa entrada. Analogamente, $M$ é polinomialmente limitada se existe um polinômio $p(n) : \mathbb{N} \longrightarrow \mathbb{N}$ tal que, para qualquer entrada $x$, o tempo consumido por $M$ é limitado superiormente por $p(|x|)$.

Por fim, se $M$ é uma MTP, valem as mesmas definições usadas para as MTNDs. Vale destacar que no caso das MTPs também aplica-se o conceito de tempo *esperado* de computação, que formalmente é a esperança do tempo consumido por uma computação de $M$ para uma dada entrada $x$.

## Espaço consumido pelas máquinas de Turing

Existem também classes de problemas que são definidas em função do espaço consumido pelas máquinas de Turing que os resolvem. Da mesma forma que no estudo do consumo de tempo, temos interesse especial nos problemas que podem ser resolvidos por máquinas de Turing que consomem espaço polinomial no tamanho da entrada.

As diferenças entre as máquinas de Turing de naturezas distintas não são muito grandes no consumo de espaço. Mais adiante, ao enunciarmos um resultado de Savitch [Sav70], ficará mais claro o porquê dessa afirmação.

Dizemos que o *espaço consumido* por uma MT é a maior quantidade de células distintas usadas pela máquina para realizar uma computação válida para uma determinada entrada. No caso das MTDs, é o número de células usadas na única computação possível. Já para MTNDs e MTPs, é o maior número de células utilizadas em uma computação válida.

Como em toda MT a cabeça de leitura só pode se deslocar de uma célula a cada passo, claramente o espaço consumido em uma computação é limitado superiormente pelo número de passos utilizados pela máquina.

Dizemos que uma MT *consome espaço polinomial* se o espaço consumido pela MT é limitado superiormente por um polinômio definido em função do tamanho da entrada.

## Linguagens e máquinas de Turing

Ao definir as máquinas de Turing, utilizamos como parâmetro um alfabeto $\Sigma$. Esse alfabeto é um conjunto que contém todos os símbolos reconhecidos pela máquina de Turing em questão. Logo, uma entrada válida para a máquina deve ser uma palavra composta apenas por símbolos de $\Sigma$.

Um conjunto de palavras cujas letras pertencem a um alfabeto $\Sigma$ é chamado de *linguagem*. Geralmente estamos interessados em verificar se uma determinada palavra pertence ou não a uma particular linguagem $L$.

Máquinas de Turing que devolvem respostas binárias podem ser usadas para efetuar essa tarefa. Uma das respostas indica que a palavra fornecida como entrada pertence à linguagem $L$ (aceitação) e a outra que a palavra não pertence (rejeição).

Em muitos casos, porém, deseja-se fornecer uma entrada para uma máquina de Turing para a obtenção de uma saída que não pode ser descrita apenas com duas respostas distintas. Por exemplo, uma máquina que recebe a representação binária de um número inteiro $x$ qualquer como entrada e devolve como resposta a representação binária do número $x + 2$ claramente deve ser capaz de devolver mais do que duas respostas distintas. Nesse caso, dizemos que estamos lidando com um *problema*, e não com uma linguagem. Rigorosamente, existem diferenças entre problemas e linguagens, mas como podemos transformar uma coisa na outra de maneira razoavelmente simples, vamos falar de problemas e linguagens de maneira indistinta. Mostraremos agora as relações entre as linguagens e as máquinas de Turing determinísticas, não-determinísticas e probabilísticas.

Se existe uma MTD $M$ que, dada uma palavra $x$, responde se $x$ pertence ou não a uma determinada linguagem $L$, então dizemos que $M$ *decide* a linguagem $L$. As respostas são devolvidas por $M$ através dos símbolos 0 e 1, que indicam rejeição e aceitação, respectivamente, da palavra $x$. Esses símbolos devem aparecer sozinhos na fita da máquina após a mesma ter parado, na segunda célula da esquerda para a direita.

No caso das MTNDs, já vimos que podem existir várias computações e várias saídas possíveis para uma máquina e uma entrada. Assim, dizemos que uma MTND $M$ decide uma linguagem $L$ se toda palavra $x$ em $L$ é aceita por $M$ em pelo menos uma de suas computações, e se toda palavra $x$ fora de $L$ é rejeitada por $M$ em todas as suas computações. Essa definição acaba mostrando a característica das MTNDs que faz com que elas pareçam mais eficientes computacionalmente e menos realistas do que as MTDs.

Por fim, a decisão de linguagens por MTPs têm um aspecto um pouco diferente das demais máquinas de Turing. Assim como as MTNDs, as MTPs também podem produzir respostas distintas para uma mesma entrada. Porém, nas MTPs, atribuímos a cada computação válida (e conseqüentemente a cada possível resposta) uma determinada probabilidade. A decisão de uma linguagem por uma MTP envolve as probabilidades de obtenção das respostas. Em alguns casos estamos interessados em fixar uma probabilidade máxima para as rejeições incorretas, em outros para as aceitações incorretas e em outros para as duas simultaneamente. A decisão de linguagens em MTPs, portanto, não possui uma definição única como no caso das MTDs e das MTNDs. Logo, ao utilizar esse conceito mais adiante no texto, definiremos exatamente o significado adequado dentro do contexto.

# Capítulo 5

# Máquinas de Turing universais

A descrição de máquinas de Turing dada no capítulo anterior indica que tais máquinas são modeladas para resolver um único problema. No entanto, os computadores atuais não estão restritos a executar uma única tarefa. Vamos descrever neste capítulo uma MTD chamada de *universal*, que tem um caráter mais geral, sendo capaz de efetuar qualquer tarefa executada por uma MTD. Depois disso, apresentaremos uma máquina de Turing quântica universal.

## 5.1 Máquina de Turing determinística universal

A descrição que apresentamos a seguir baseia-se na apresentação de Papadimitriou [Pap94]. Denotaremos por $U$ a máquina de Turing universal que vamos construir. Essa máquina lê uma entrada e a interpreta como sendo a descrição de uma máquina de Turing $M$ e uma entrada $x$ para essa máquina. A máquina $U$ deve funcionar de modo que, após sua parada, a saída produzida seja a mesma que $M$ devolve ao ser executada tendo $x$ como entrada.

Vamos determinar agora o padrão de entrada a ser reconhecido por $U$. Sejam $M = (Q, \Sigma, \delta, s, H)$ a máquina que será simulada e $x$ sua entrada. Cada símbolo de $\Sigma$ e cada estado de $Q$ será descrito como um número na forma binária. Os símbolos de $\Sigma$ serão representados por números em $\{1, 2, \ldots, |\Sigma|\}$, os estados de $Q$ por números em $\{|\Sigma| + 1, |\Sigma| + 2, \ldots, |\Sigma| + |Q|\}$ e os números $|\Sigma| + |Q| + 1, \ldots, |\Sigma| + |Q| + 5$ representarão os símbolos $\{\leftarrow, \downarrow, \rightarrow\}$, $\triangleright$ e $\sqcup$. Todos os símbolos serão representados usando a mesma quantidade de bits (serão usados zeros à esquerda quando necessário). O estado denotado por $s$ é o estado inicial de $M$, e será representado pelo número $|\Sigma| + 1$.

Na parte da entrada que corresponde a $M$, inicialmente serão fornecidos os números $|Q|$, $|\Sigma|$ e $|H|$. Em seguida, aparece a representação de $|H|$ estados de

$M$, indicando seus estados finais. Por fim, segue uma descrição de $\delta$ através de uma seqüência de pares na forma $((q,\sigma),(p,\rho,D))$, onde cada símbolo estará codificado na forma já descrita. Vamos assumir sem perda de generalidade que os parênteses e a vírgula pertencem ao alfabeto de $U$.

Após a descrição da máquina, a fita de entrada de $U$ vai conter o símbolo ";" (que deverá pertencer ao alfabeto de $U$), seguido pela descrição da entrada $x$ para $M$, também devidamente codificada. Como cada símbolo e estado é representado com o mesmo número de bits, não há ambigüidades na descrição da máquina ou da entrada.

Tendo a entrada, $U$ pode começar sua execução. Para facilitar a descrição, vamos supor que $U$ utiliza 3 fitas, e que cada fita tem uma cabeça de leitura independente (ou seja, cada cabeça pode estar numa posição diferente das demais). Pode-se provar que uma MTD com uma fita pode simular qualquer MTD com $k$ fitas e com $k$ cabeças de leitura independentes para $k > 1$.

Na primeira fita, $U$ receberá a entrada e devolverá a saída. Na segunda fita, $U$ vai guardar a configuração atual de $M$. A configuração estará no formato $(w,q,u)$, com cada símbolo codificado da maneira que descrevemos anteriormente. Inicialmente, a configuração inicial será $(\triangleright, s, x)$. A terceira fita guardará a descrição da função de transição $\delta$ de $M$.

Para simular $M$, no início de cada passo $U$ percorre a segunda fita até descobrir o estado atual de $M$, localizando assim a posição da cabeça de leitura de $M$ em sua fita. Em seguida, $U$ percorre a terceira fita, procurando uma transição que tenha $q$ como estado do domínio, onde $q$ é o estado atual. Feito isso, $U$ move a cabeça de leitura para a esquerda na segunda fita para identificar o símbolo $\sigma$ que está sob a cabeça de leitura de $M$ e verificar se a transição que está sendo analisada atualmente tem como domínio o par $(q,\sigma)$. Se não for esse o par, repete-se o processo até que ele seja encontrado.

Após localizar a transição adequada, $U$ faz as modificações na segunda fita, movimentando a cabeça, alterando o símbolo na posição que estava anteriormente e mudando o estado atual, conforme a descrição da função de transição. Se o estado atingido pertence a $H$, a simulação de $M$ pára. Caso contrário o processo se repete.

Ao término da simulação, o conteúdo da segunda fita de $U$ representará a configuração final de $M$ após sua execução para a entrada $x$. Assim, $U$ substitui o conteúdo da primeira fita pelo da segunda e termina sua execução.

A construção dessa máquina utilizou uma série de recursos e ferramentas que facilitaram bastante nossa tarefa, como as fitas múltiplas e o movimento independente das cabeças de leitura dessas fitas. Nem todas essas ferramentas podem ser utilizadas no modelo quântico. Além disso, outros cuidados devem ser tomados, e isso acaba resultando num aumento significativo da complexidade da correspondente construção.

A partir da descrição mostrada, não é difícil deduzir que uma MTD pode simular também uma MTND ou uma MTP. Vale destacar, porém, que no caso de MTNDs ou MTPs, alguns detalhes e convenções devem ser ajustados pois essas máquinas podem produzir mais de uma saída para a mesma entrada.

## 5.2 Máquina de Turing quântica universal

As operações realizadas por uma MTQ podem ser descritas como transformações unitárias sobre vetores de um espaço de Hilbert. Logo, o estudo dessas transformações facilita a compreensão das limitações e das ferramentas disponíveis quando estudamos a computação quântica.

Nesta seção, apresentamos alguns resultados de Bernstein e Vazirani [BV97]. Mostramos como uma dada tranformação unitária pode ser decomposta em transformações unitárias simples (ou *quase-triviais*), diretamente implementáveis em MTQs. Essas transformações simples são divididas em dois tipos: *mudança de fase* e *rotação*.

O resultado apresentado é construtivo, mas envolve cálculos com números irracionais. A princípio, não consideramos as precisões das operações envolvidas, mas vamos mostrar como os ângulos utilizados nas operações de rotação e de mudança de fase podem ser obtidos com precisão $\epsilon$ consumindo tempo polinomial em $1/\epsilon$ a partir de um ângulo fixo.

### 5.2.1 Decomposição de uma transformação unitária

Se uma MTQ com conjunto de estados $Q$ e alfabeto $\Sigma$, durante uma computação, utiliza no máximo $k$ células da fita, então ela admite que uma matriz unitária de dimensão $k|Q||\Sigma|^k$ descreva sua evolução. Logo, a dimensão do espaço considerado será $d$, onde $d = k|Q||\Sigma|^k$. Denotamos por $e_i$ o vetor com todas as coordenadas nulas exceto a coordenada $i$, que vale 1.

Uma matriz unitária $M$ $d \times d$ é denominada *quase-trivial* se satisfaz uma das duas condições abaixo:

1. $M$ é a matriz identidade a menos de uma de suas entradas na diagonal principal, que tem valor $e^{i\theta}$ para algum $\theta \in [0, 2\pi)$. Ou seja, existe j tal que $M_{j,j} = e^{i\theta}$, $M_{k,k} = 1$ $\forall k \neq j$ e $M_{k,l} = 0$ $\forall k, l,$ $k \neq l$.

2. $M$ é a identidade a menos da submatriz $2 \times 2$ determinada por um par de coordenadas distintas $j$ e $k$, onde coincide com a rotação de um ângulo

$\theta \in [0, 2\pi)$, dada pela matriz: $\begin{pmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{pmatrix}$. Ou seja, $M$ é tal que existem $\theta$ e $j \neq k$ tais que $Me_j = (\cos\theta)e_j + (\operatorname{sen}\theta)e_k$, $Me_k = -(\operatorname{sen}\theta)e_j + (\cos\theta)e_k$, e para $\forall l \neq j, k, Me_l = e_l$. Nesse ponto, nos referimos a $Me_l$ como sendo a $l$-ésima coluna de $M$.

Dizemos que uma transformação que satisfaz a primeira condição é uma *mudança de fase* quase-trivial, enquanto que uma transformação que satisfaz a segunda condição é uma *rotação* quase-trivial.

Vamos usar a notação $[j, j, \theta]$ para representar uma mudança de fase de $e^{i\theta}$ na coordenada $j$, e vamos usar a notação $[j, k, \theta]$ para denotar uma rotação de um ângulo $\theta$ entre as coordenadas $j$ e $k$.

Agora, mostramos uma tranformação que leva um vetor $v$ em $\mathbb{C}^d$ em um vetor de mesmo módulo mas com apenas uma coordenada específica não-nula. Esse resultado será utilizado no teorema que mostra a decomposição de uma matriz unitária em matrizes quase-triviais. Para isso, vamos definir duas funções sobre o conjunto dos números complexos. Se $x \in \mathbb{C}$, $\mathrm{Re}(x)$ é o valor da parte real de $x$, e $\mathrm{Im}(x)$ é o valor da parte imaginária de $x$.

**Lema 5.1.** *Para todo vetor $v \in \mathbb{C}^d$, existe um conjunto de $2d-1$ matrizes quase-triviais $U_1, U_2, \ldots, U_{2d-1}$ tais que*

$$U_1 U_2 \ldots U_{2d-1} v = \|v\| e_1.$$

*Demonstração.* Inicialmente, transformamos o vetor $v = (v_1, \ldots, v_d)$ em um vetor de $\mathbb{R}^d$ utilizando mudanças de fase.

Seja $P_j = [j, j, \phi_j]$ a matriz quase-trivial que, aplicada na coordenada $j$, faz uma mudança de fase de um ângulo $\phi_j$ onde $\phi_j = 0$ se $v_j = 0$ ou $\phi_j = 2\pi - \arccos \frac{\mathrm{Re}(v_j)}{|v_j|}$ ou $\phi_j = \arccos \frac{\mathrm{Re}(v_j)}{|v_j|}$, dependendo do sinal de $\mathrm{Im}(v_j)$. Assim, temos $v_j = |v_j| e^{i\theta} = |v_j|(\cos\theta + i \operatorname{sen}\theta)$ para algum $\theta$, e aplicando $P_j$ em $v_j$ obtemos um vetor $v_j' = e^{i\theta'}$ tal que $\operatorname{sen}\theta' = 0$.

Feitas as mudanças de fase em todas as coordenadas, teremos um novo vetor $v' = P_1 \cdots P_d v$, com $v_j' = |v_j|$ para todo $j$.

O próximo passo é fazer $d-1$ rotações para mover todo o peso do vetor $v'$ para a sua primeira coordenada. Assim, seja $R_j = [j, j+1, \theta_j]$ a matriz quase-trivial que aplica nas coordenadas $j$ e $j+1$ uma rotação de $\theta_j$, onde

$$\theta_j = -\arccos \frac{|v_j|}{\sqrt{\sum_{k=j}^{d} |v_k|^2}}$$

se a somatória no denominador for não-nula e $\theta_j = 0$ caso contrário. Dessa forma, temos que

$$R_1 \cdots R_{d-1} P_1 \cdots P_d v = \|v\| e_1.$$

Na rotação $R_j$, a $(j+1)$-ésima e a $j$-ésima coordenadas serão alteradas, de modo que a primeira ficará com valor 0 e a segunda vai concentrar o peso que as duas tinham anteriormente. Como o vetor em questão tem dimensão $d$, serão necessárias exatamente $d-1$ rotações para que o vetor resultante tenha todo o seu peso concentrado em sua primeira coordenada.

$\square$

Para facilitar a compreensão do lema acima, vejamos um exemplo.

**Exemplo 5.2** Seja $v = (1+i, i, 2)$ um vetor em $\mathbb{C}^3$. Vamos mapear esse vetor em $\mathbb{R}^3$. Para isso, serão necessárias três mudanças de fase:

1. Mudança na terceira coordenada:

   Em primeiro lugar, devemos obter o valor de $\phi_3$. Como $v_3 = 2$, temos:

   $$\phi_3 = \arccos \frac{\mathrm{Re}(v_3)}{|v_3|} = \arccos \frac{2}{2} = 0.$$

   Logo, temos que $P_3$ é a transformação quase-trivial $[3, 3, 0]$, e portanto $P_3 v = (1+i, i, 2)$.

2. Mudança na segunda coordenada:

   Agora, devemos obter o valor de $\phi_2$. Como $v_2 = i$, temos:

   $$\phi_2 = 2\pi - \arccos \frac{\mathrm{Re}(v_2)}{|v_2|} = 2\pi - \arccos \frac{0}{1} = 2\pi - \frac{\pi}{2} = \frac{3\pi}{2}.$$

   Logo, temos que $P_2$ é a transformação quase-trivial $\left[2, 2, \frac{3\pi}{2}\right]$. Assim, como $P_3 v = v$, temos:

   $$e^{\frac{3\pi i}{2}} v_2 = e^{\frac{3\pi i}{2}} e^{\frac{\pi i}{2}} = e^{2\pi i} = \cos 2\pi + i \operatorname{sen} 2\pi = 1$$

   e portanto $P_2 P_3 v = (1+i, 1, 2)$.

3. Mudança na primeira coordenada:

   Por fim, fazemos a conversão da primeira coordenada. O esquema é parecido com a conversão da terceira coordenada. Sendo $v_1 = 1+i$, temos:

   $$\phi_1 = 2\pi - \arccos \frac{\mathrm{Re}(v_1)}{|v_1|} = 2\pi - \arccos \frac{1}{\sqrt{2}} = 2\pi - \frac{\pi}{4} = \frac{7\pi}{4}.$$

   Logo, temos que $P_1$ é a transformação quase-trivial $\left[1, 1, \frac{7\pi}{4}\right]$. Assim:

$$e^{\frac{7\pi i}{4}} v_1 = \sqrt{2}\ e^{\frac{7\pi i}{4}} e^{\frac{\pi i}{4}} = \sqrt{2}\ e^{2\pi i} = \sqrt{2}\ (\cos 2\pi + i \operatorname{sen} 2\pi) = \sqrt{2}$$

e portanto $P_1 P_2 P_3 v = (\sqrt{2}, 1, 2)$.

Dessa forma, mapeamos $v \in \mathbb{C}^3$ em $\mathbb{R}^3$. Agora, vamos fazer duas rotações para mover todo o peso do vetor $P_1 P_2 P_3 v$ para sua primeira coordenada.

1. Rotação entre a $3^{\underline{a}}$ e a $2^{\underline{a}}$ coordenada:

   Nesse primeiro passo, vamos mover todo o peso da $3^{\underline{a}}$ e da $2^{\underline{a}}$ coordenadas do vetor $P_1 P_2 P_3 v = (\sqrt{2}, 1, 2)$ para a segunda coordenada. Para isso, vamos aplicar a rotação $R_2 = [2, 3, \theta_2]$ onde

   $$\theta_2 = -\arccos \frac{|v_2|}{\sqrt{\sum_{j=2}^{3} |v_j|^2}} = -\arccos \frac{1}{\sqrt{5}}.$$

   Deixamos a cargo do leitor verificar que a multiplicação da matriz que corresponde a $R_2$ por $(\sqrt{2}, 1, 2)$ resulta no vetor $(\sqrt{2}, \sqrt{5}, 0)$.

2. Rotação entre a $2^{\underline{a}}$ e a $1^{\underline{a}}$ coordenada:

   Nesse último passo, movemos todo o peso de $R_2 P_1 P_2 P_3 v = (\sqrt{2}, \sqrt{5}, 0)$ para a primeira coordenada. Vamos aplicar a rotação $R_1 = [1, 2, \theta_1]$, onde

   $$\theta_1 = -\arccos \frac{|v_1|}{\sqrt{\sum_{j=1}^{3} |v_j|^2}} = -\arccos \sqrt{\frac{2}{5}}.$$

   Mais uma vez, deixamos as contas a cargo do leitor, e afirmamos por fim que a multiplicação da matriz correspondente a $R_1$ por $(\sqrt{2}, \sqrt{5}, 0)$ resulta no vetor $(\sqrt{7}, 0, 0)$.

Note que $R_1 R_2 P_1 P_2 P_3 v = \|v\| e_1$, conforme enunciado no lema.

Vamos mostrar agora como o mapeamento de um vetor para uma determinada direção permite a decomposição de uma transformação unitária.

**Teorema 5.3.** *Para toda matriz unitária $U$ de dimensão $d$, existem $n$ matrizes quase-triviais $U_1, \ldots, U_n$ de dimensão $d$, com $n$ limitado por um polinômio em $d$, tais que $U = U_n \cdots U_1$.*

*Demonstração.* Para determinar as matrizes $U_1, \ldots, U_n$, vamos determinar matrizes $Q_1, \ldots, Q_d$ tais que $Q_d \cdots Q_1 U$ é a matriz identidade, onde cada matriz $Q_k$ pode ser facilmente convertida num produto de matrizes quase-triviais. Como a inversa de uma matriz quase-trivial também é quase-trivial, de $Q_1, \ldots, Q_d$ é fácil obter as matrizes $U_1, \ldots, U_n$ mencionadas no enunciado do teorema.

Uma matriz $d \times d$ é *k-simples* se suas primeiras $k$ linhas e suas primeiras $k$ colunas são iguais as da identidade. A única matriz $d \times d$ $d$-simples é a identidade. É claro que o produto de duas matrizes $k$-simples também é uma matriz $k$-simples.

As matrizes $Q_1, \ldots, Q_d$ são obtidas seqüencialmente, de modo que $Q_k \cdots Q_1 U$ é uma matriz $k$-simples para $k = 1, \ldots, d$ (isso implica que $Q_d \cdots Q_1 U$ é a identidade).

Suponha que a matriz $S = Q_{k-1} \cdots Q_1 U$ é uma matriz $(k-1)$-simples unitária. Queremos um produto de matrizes quase-triviais $Q_k$ tal que $Q_k S$ seja $k$-simples. Para obter a matriz $Q_k$, vamos recorrer ao resultado mostrado no lema anterior.

Seja $Z$ a matriz obtida de $S$ removendo-se as $k-1$ primeiras linhas e colunas de $S$. Note que $Z$ é uma matriz $(d-k+1)$-dimensional. Vamos aplicar a transformação do lema anterior no vetor correspondente à primeira linha de $Z$, que denotaremos por $Z_1^T$. A saída fornecida pelo algoritmo será uma seqüência de matrizes unitárias quase-triviais $V_1, V_2, \ldots, V_{2d-2k+1}$ $(d-k+1)$-dimensionais tais que $V_1 \cdots V_{2d-2k+1} Z_1^T = \|Z_1^T\| e_1$.

Feito isso, estendemos cada $V_i$ para transformá-la em uma matriz $(k-1)$-simples $d$-dimensional. Como $S$ é unitária, temos que $Q_k S$ é uma matriz $k$-simples unitária.

A cada passo, produzimos uma das matrizes $Q_k$. Cada uma dessas matrizes é um produto de matrizes $d$-dimensionais quase-triviais. No final, a matriz $Q_d \cdots Q_1 U$ será a identidade de dimensão $k$. A partir do produto de matrizes quase-triviais que compõem as matrizes $Q_k$, podemos obter as matrizes $U_j$ do enunciado.

□

Bernstein e Vazirani [BV97] apresentaram versões algorítmicas do lema 5.1 e do teorema 5.3. Observe que as provas do lema e do teorema são construtivas se ignorarmos o fato de que envolvem cálculos com números reais. Bernstein e Vazirani demonstraram variantes "aproximadas" destes resultados. Tais variantes mostram que, mesmo que os cálculos envolvidos sejam feitos com um certo erro de precisão, as conclusões, ajustadas adequadamente, valem.

### 5.2.2 Cálculo de transformações quase-triviais

Agora, apresentamos um lema que mostra que podemos usar uma única rotação para aproximar eficientemente o valor de qualquer rotação. Ou seja, podemos aproximar qualquer ângulo a partir de um determinado ângulo fixo.

**Lema 5.4.** *Seja $\mathcal{R} = 2\pi \sum_{i=1}^{\infty} 2^{-2^i}$. Então existe um algoritmo determinístico que, dados $\epsilon > 0$ e um ângulo $\theta \in [0, 2\pi)$, produz um inteiro $k$ limitado polinomialmente por $1/\epsilon$ tal que*

$$|k\mathcal{R} - \theta| \mod 2\pi \leq \epsilon.$$

*Demonstração.* Seja $n$ a menor potência de 2 tal que $\epsilon > \frac{2\pi}{2^{n-1}}$. Aproximamos o número $\frac{\theta}{2\pi}$ por uma fração de denominador $2^n$. Ou seja, encontramos um inteiro $m \in [1, 2^n)$ tal que

$$\left|\frac{\theta}{2\pi} - \frac{m}{2^n}\right| \leq \frac{1}{2^n}.$$

Dessa forma, podemos pegar $k = m2^n$, pois

$$\begin{aligned}
m2^n\mathcal{R} \mod 2\pi &= \left(2\pi m \sum_{i=1}^{\infty} 2^{n-2^i}\right) \mod 2\pi \\
&= \left(2\pi m \sum_{i=\log n+1}^{\infty} 2^{n-2^i}\right) \mod 2\pi \\
&= \left(\frac{2\pi m}{2^n} + 2\pi m \sum_{i=\log n+2}^{\infty} 2^{n-2^i}\right) \mod 2\pi,
\end{aligned}$$

e como

$$m \sum_{i=\log n+2}^{\infty} 2^{n-2^i} \leq m2^{n-4n+1} \leq 2^{n-3n+1} \leq 2^{-2n+1},$$

temos que

$$\begin{aligned}
|m2^n\mathcal{R} - \theta| \mod 2\pi &\leq \left|m2^n\mathcal{R} - \frac{2\pi m}{2^n}\right| \mod 2\pi + \left|\frac{2\pi m}{2^n} - \theta\right| \\
&\leq \frac{2\pi}{2^{2n-1}} + \frac{2\pi}{2^n} < \frac{2\pi}{2^{n-1}} < \epsilon.
\end{aligned}$$

□

O resultado acima, para ser utilizado em combinação com o teorema 5.3 apresentado nessa seção, precisa de uma adaptação, bem como o lema 5.1. Há ainda a necessidade de admitir que, em potenciais implementações de MTQs e na execução das transições, eventualmente aparecem erros de precisão. Bernstein e Vazirani [BV97] apresentam versões adaptadas do lema e do teorema que levam em consideração potenciais erros de precisão e a utilização da aproximação dada pelo último lema.

A utilização das decomposições em resultados algorítmicos, como por exemplo na descrição de uma MTQ universal, envolvem ainda a questão de se tais decomposições aproximadas podem ser obtidas em tempo polinomial nos parâmetros envolvidos. A versão de Bernstein e Vazirani do teorema incorpora esses dois aspectos e é enunciado abaixo, após uma definição, para que fiquem um pouco mais claras as conseqüências do teorema.

Uma matriz $U$ é *$\epsilon$-próxima de unitária* se existe uma matriz unitária $\tilde{U}$ tal que $\|U - \tilde{U}\| \leq \epsilon$.

**Teorema 5.5 (Bernstein & Vazirani).** *Existe um algoritmo determinístico que tem como entrada um número real $\epsilon > 0$ e uma matriz $d \times d$ complexa $\frac{\epsilon}{2(10\sqrt{d})^d}$-próxima de unitária e que produz matrizes quase-triviais $d$-dimensionais $U_1, \ldots, U_n$, com $n$ polinomial em $d$, tais que $\|U - U_n \cdots U_1\| \leq \epsilon$. O algoritmo consome tempo polinomial em $\log 1/\epsilon$ e no tamanho de $U$.*

### 5.2.3 Descrição da máquina de Turing quântica universal

Como na teoria clássica de complexidade, para apresentar uma MTQ universal, é necessário introduzir ferramentas que permitam a simulação de uma MTQ arbitrária por uma MTQ universal. Esse ferramental, apesar de técnico e importante, não apresenta grandes dificuldades, até por não envolver nada peculiar ao modelo quântico de computação. Por isso, optamos por omitir sua apresentação, concentrando-nos na apresentação dos conceitos principais.

Na descrição da MTD universal, utilizamos uma máquina com várias fitas. Fitas múltiplas, porém, são de difícil trato no modelo quântico, e foram substituídas por uma fita única com múltiplas "trilhas", de efeito quase semelhante.

Uma diferença está no fato de que, por ter uma única fita, a MTQ universal tem apenas uma cabeça de leitura. Ou seja, a MTQ universal lê todas as trilhas em uma mesma posição simultaneamente, ao contrário da MTD universal, que utilizava uma cabeça para cada fita. Esse ponto e a questão de sincronização, discutida a seguir, dificultam a simulação de uma MTQ e justificam o funcionamento da MTQ universal descrita na prova do próximo teorema.

Para que uma MTQ universal possa simular as transições quânticas de uma

MTQ $M$, ela deve se manter sincronizada, ou seja, todas as configurações da superposição de $M$ num determinado passo devem ser atingidas simultaneamente (não deve haver transições atrasadas ou adiantadas na simulação).

A simulação realizada por uma MTQ universal não é exata, ao contrário do que acontece com a MTD universal. Alguns desses detalhes serão omitidos para que não prejudiquem a compreensão das idéias principais envolvidas na simulação.

Para descrever a MTQ universal, vamos precisar do conceito de *máquina unidirecional*, que definimos a seguir.

Uma MTQ $M = (Q, \Sigma, \alpha, s, H)$ é denominada *unidirecional* se cada estado pode ser acessado a partir de uma única direção. Em outras palavras, se $\alpha(p_1, q_1, p'_1, q, d_1)$ e $\alpha(p_2, q_2, p_2 1, q, d_2)$ são ambas não-nulas, então $d_1 = d_2$.

Bernstein e Vazirani [BV97] demonstraram que toda MTQ pode ser simulada por uma MTQ unidirecional em um número de passos no máximo 5 vezes maior que o da MTQ simulada. Mais do que isso, a descrição da MTQ unidirecional pode ser obtida da descrição da MTQ a ser simulada em tempo polinomial.

Esse fato é usado na prova do teorema abaixo. Nesse texto, apresentamos apenas a idéia geral dessa prova, que envolve uma série de detalhes técnicos, que podem ser encontrado no artigo de Bernstein e Vazirani [BV97].

**Teorema 5.6.** *Existe uma MTQ $U$ que, dada a descrição de uma MTQ $M$, um número $\epsilon > 0$ e um inteiro $T \geq 0$, simula os $T$ primeiros passos de $M$ com com precisão $\epsilon$ em um número de passos polinomial em $T$ e $1/\epsilon$.*

*Idéia da demonstração:* Seja $U$ nossa MTQ universal, e seja $M$ a máquina que queremos simular. Sabemos que existe uma MTQ $M' = (Q, \Sigma, \alpha, s, H)$ unidirecional que simula $M$ consumindo tempo polinomial no número de passos de $M$. Vamos usar $U$ para simular $M'$.

As configurações de $M'$ são guardadas em duas trilha da fita de $U$. Para simular $M'$, serão utilizadas $\log |Q \times \Sigma|$ células de $U$ por célula de $M'$. Cada um desses grupos de células contém um par ordenado $(p, \sigma)$, onde $\sigma \in [1, |\Sigma|]$ representa o conteúdo da célula correspondente em $M'$, e $p \in [0, |Q|]$ representa o estado de $M'$ se a cabeça está nessa célula e $p = 0$ se a cabeça estiver em outra célula de $M'$. Cada membro do par estará em duas trilhas da fita. Seja $T$ o número de passos de $M$. O número de grupos de células utilizados para simular a fita de $M'$ será $2T + 1$.

Sabemos que, ignorado o movimento da cabeça de leitura, $\alpha$ fornece uma transformação unitária $V$ de dimensão $d = |Q \times \Sigma|$. Para fazer a atualização da superposição de $M'$ armazenada nas duas trilhas de $U$, inicialmente varremos a fita de $U$ e fazemos uma cópia do estado corrente e do símbolo sob a cabeça de leitura de $M'$ para um grupo de células fixo de $U$, especificamente reservado para isso.

Com a cabeça de leitura de $U$ sob esse grupo de células, aplicamos a transformação $V$, obtendo o novo estado e o novo símbolo sob a cabeça de leitura de $M'$ para cada uma das configurações de sua superposição corrente. De posse dessa informação, $U$ varre novamente sua fita, atualizando o estado e o símbolo na posição correta de sua fita. A partir do estado corrente, dado que $M'$ é uma máquina unidirecional, $U$ determina o movimento da cabeça de leitura de $M'$.

Na verdade, a partir de $\alpha$, podemos obter, em tempo polinomial, apenas uma aproximaçaão de $V$ com erro $\epsilon$.

A necessidade de se fazer uma cópia do estado e do símbolo sob a cabeça de leitura de $M'$ vem do fato de que a cabeça de leitura das várias configurações de $M'$ não estão na mesma posição e a transição de $U$ deve ser feita simultaneamente em todas essas configurações.

# Capítulo 6

# Classes de complexidade

## 6.1 Classes de complexidade do modelo clássico

### As classes P e PSPACE

A partir do conceito de máquinas de Turing polinomialmente limitadas, vamos definir as linguagens *polinomialmente decidíveis*. Dizemos que uma linguagem é polinomialmente decidível se existe uma máquina de Turing determinística polinomialmente limitada que a decide.

A classe **P** (*polynomial-time*) é o conjunto de todas as linguagens polinomialmente decidíveis. Esta classe é extremamente importante, pois contém a maior parte dos problemas que podem ser resolvidos de maneira eficiente. Dizemos que um problema é resolvido de maneira eficiente se existe um algoritmo para resolvê-lo que consome tempo polinomial no tamanho da entrada. A classe **P** é fechada sob união, intersecção, concatenação e estrela de Kleene. Essas propriedades são importantes e suas provas são interessantes, mas serão omitidas nesse texto.

De maneira parecida, podemos definir a classe **PSPACE** (*polynomial-space*). Vale ressaltar que, enquanto a primeira faz a classificação das linguagens em função do tempo consumido, a segunda o faz em função do espaço consumido. Dizemos que uma linguagem pertence à classe **PSPACE** se ela é decidida por uma máquina de Turing determinística que consome espaço polinomial no tamanho da entrada.

É fácil ver que $\mathbf{P} \subseteq \mathbf{PSPACE}$: como já observamos, toda MTD que consome tempo polinomial certamente consome espaço polinomial, já que, a cada passo, a cabeça de leitura da máquina só pode se mover de no máximo uma célula à direita. Por outro lado, o inverso não é verdade. É fácil imaginar uma MTD que consome tempo superpolinomial mas espaço polinomial. Assim, é concebível (e até de se esperar) que existam linguagens em **PSPACE** que não estejam em **P**. Surpreendentemente, não se sabe até hoje se este é o caso ou não. Ou seja, não se sabe se $\mathbf{P} = \mathbf{PSPACE}$.

## As classes NP e coNP

As classes **NP** e **coNP** podem ser descritas de maneira parecida com a que descrevemos a classe **P**. Conforme explicaremos adiante, essas classes são muito importantes no estudo da teoria de complexidade clássica.

Vamos definir **NP** (*nondeterministic polynomial-time*) e **coNP** em função das máquinas de Turing não-determinísticas polinomialmente limitadas. Dizemos que uma linguagem pertence à classe **NP** se ela pode ser decidida por uma máquina de Turing não-determinística polinomialmente limitada.

A classe das linguagens cujo complemento pertence a **NP** é denominada **coNP**, onde o complemento de uma linguagem $L$ sob um alfabeto $\Sigma$ é a linguagem $\Sigma^* \setminus L$. De maneira geral, o complemento de uma classe de complexidade arbitrária **C** é denotado por **coC** e definido como o conjunto das linguagens cujo complemento está em **C**.

Por exemplo, ao definir a classe **P**, também podemos definir a classe **coP**, seguindo o esquema acima. Nesse caso, temos uma classe que é igual ao seu complemento. Dada uma linguagem $L$ em **P** ou em **coP**, e uma máquina $M$ polinomialmente limitada que a decide, basta trocar aceitação por rejeição e vice-versa na descrição de $M$ para obter uma máquina polinomialmente limitada que decide o complemento de $L$.

As classes **NP** e **coNP** contêm vários problemas para os quais não se conhece algoritmo polinomial, e formam com a classe **P** a fronteira entre o que pode e o que não pode ser resolvido eficientemente no modelo clássico. Essas classes também são importantes porque contêm uma grande quantidade de problemas de interesse prático.

Claramente, toda linguagem que está em **P** também está em **NP** e em **coNP**, visto que uma MTD é uma MTND. Porém, não sabemos muito mais a respeito da relação entre essas três classes. A questão mais importante e conhecida é se **P** é igual a **NP**. Se isso for verdade, saberemos que muitos problemas para os quais hoje não se conhecem algoritmos exatos razoáveis terão uma solução polinomial. Porém, são poucos os que acreditam nessa hipótese.

## $\mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE}$

Uma relação conhecida entre classes de complexidade é que $\mathbf{NP} \subseteq \mathbf{PSPACE}$. Isso significa que toda linguagem decidida por uma MTND limitada polinomialmente pode ser decidida por uma MTD que consome espaço polinomial.

Vamos apresentar a prova de maneira razoavelmente informal, pois o que queremos mostrar é a idéia que está por trás da simulação de uma MTND por uma MTD.

Dada uma MTND $M$ polinomialmente limitada que decide uma linguagem $L$, queremos mostrar que essa máquina pode ser simulada por uma MTD $M'$ que consome espaço polinomial e decide a mesma linguagem $L$.

A diferença entre as máquinas $M$ e $M'$ é o não-determinismo. Sendo uma MTND, $M$ requer apenas que uma de suas possíveis computações aceite uma palavra da linguagem $L$, e também requer que todas as computações rejeitem as palavras que não estão em $L$.

Logo, ao simular $M$, a máquina $M'$ deve ser capaz de simular todas as computações possíveis de $M$ para poder rejeitar uma palavra corretamente. A conclusão de que uma palavra está na linguagem pode ser rápida (basta encontrar a primeira computação que aceita a palavra de entrada), mas em geral não é possível determinar a priori quantas simulações precisam ser feitas.

Assim, considerando o pior caso, é fácil ver que toda simulação de $M$ por $M'$ que consiste em testar cada uma das computações possíveis pode consumir tempo exponencial no tamanho da entrada. Porém, sabemos que $M$ é polinomialmente limitada. Logo, todas as suas computações consomem tempo, e portanto espaço, polinomial no tamanho da entrada.

Assim, uma simulação de $M$ por $M'$ que testa cada computação possível consumindo espaço polinomial e que sempre consegue reaproveitar esse espaço nas próximas computações é suficiente para mostrar que **NP** está contida em **PSPACE**.

A construção da $M'$ é razoavelmente simples. A idéia principal consiste numa espécie de busca em largura num grafo. Mais especificamente, $M'$ simula todas as computações possíveis de $M$ com $t$ passos antes de simular qualquer computação com $t + 1$ passos.

Para representar as computações, $M'$ pode indexar as transições das computações com números. Esses números serão obtidos a partir de um contador, que será representado na base binária. Logo, mesmo se o número de computações for exponencial, o espaço consumido por $M'$ para armazenar tal contador será polinomial no tamanho da entrada.

Em cada passo da simulação, $M'$ incrementa seu contador e simula a computação de $M$ referente a seu valor. O conteúdo da fita de $M$ para cada computação pode ser guardado numa fita auxiliar durante a simulação, e antes da próxima computação essa fita é apagada. Dessa forma, é claro que o espaço usado para guardar as computações também será polinomial no tamanho da entrada.

Como toda transição de $M$ pode ser simulada em tempo polinomial por $M'$, a simulação pode ser feita consumindo espaço polinomial no tamanho da entrada. Assim, **NP** $\subseteq$ **PSPACE**. Analogamente, mostra-se que **coNP** $\subseteq$ **PSPACE**.

Dizemos que uma linguagem pertence à classe **NPSPACE** se ela é decidida por uma máquina de Turing não-determinística que consome espaço polinomial no tamanho da entrada. Uma simulação semelhante a essa foi proposta por Savitch [Sav70] para mostrar que **NPSPACE** = **PSPACE**.

## As classes RP e coRP

As classes discutidas a seguir envolvem computações probabilísticas e são de fundamental importância tanto no modelo clássico como no modelo quântico.

Primeiro definimos a classe **RP** (*randomized polynomial-time*). Na literatura, essa classe é definida em função das *máquinas de Turing Monte-Carlo*. Infelizmente não há um consenso quanto a definição dessas máquinas (algumas fontes falam que elas devem ser limitadas polinomialmente, enquanto outras não estabelecem essa restrição). Logo, vamos fazer a definição da classe em função de suas propriedades principais.

Para que uma linguagem $L$ pertença à classe **RP**, deve existir uma MTP $M$ limitada polinomialmente satisfazendo o seguinte:

1. Se uma palavra $x$ dada na entrada não está na linguagem $L$, então a máquina $M$ sempre rejeita $x$.

2. Se uma palavra $x$ dada na entrada está na linguagem $L$, então a máquina $M$ aceita $x$ com probabilidade maior ou igual a 0,5.

Essas duas propriedades caracterizam a máquina de Turing Monte-Carlo. Diz-se neste caso que M decide a linguagem L com probabilidade de aceitação 0,5.

O valor da probabilidade de aceitação na definição de **RP** não é muito importante, no sentido de que qualquer outro valor em $(0;1]$ leva à mesma classe **RP**. Isso porque, de uma MTP limitada polinomialmente que decide uma linguagem $L$ com probabilidade de aceitação $p$, para um valor $p$ qualquer em $(0,1]$, é possível obter uma MTP limitada polinomialmente que decide $L$ com probabilidade de aceitação 0,5. De fato, se $p \geq 0{,}5$, não há nada a fazer. Se $p < 0{,}5$, executa-se $k$ vezes, com $k = \lceil -1/\log p \rceil$, a máquina original e aceita-se a palavra se pelo menos uma vez a máquina original a aceitou. Do contrário, rejeita-se a palavra. Esse procedimento aumenta a probabilidade de aceitação para pelo menos 0,5.

O uso da repetição da execução das máquinas de Turing Monte-Carlo limitadas polinomialmente é bastante útil, pois pode deixar a probabilidade de falsas rejeições arbitrariamente pequena. Por isso, esse recurso é bastante utilizado na prática, mesmo quando as probabilidades de rejeição incorreta são pequenas (menores que 0,5).

A classe **coRP**, das linguagens cujo complemento está em **RP**, pode também ser vista como a classe das linguagens $L$ para as quais existe uma MTP $M$ limitada polinomialmente com as seguintes propriedades:

1. Se uma palavra $x$ dada na entrada não está na linguagem $L$, então a máquina $M$ rejeita $x$ com probabilidade maior ou igual a 0,5.

2. Se uma palavra $x$ dada na entrada está na linguagem $L$, então a máquina $M$ nunca rejeita a palavra $x$.

As definições deixam claro o caráter complementar das duas classes. Enquanto as linguagens que estão em **RP** admitem máquinas que fazem rejeições incorretas, as linguagens de **coRP** admitem máquinas que fazem aceitações incorretas.

A classe **RP** claramente está contida em **NP**. Basta notar que uma máquina de Turing Monte-Carlo para uma linguagem $L$ é uma máquina de Turing não-determinística que decide $L$ (existe pelo menos uma computação dessa máquina que aceita cada palavra que está em $L$ e todas as computações rejeitam palavras que não estão em L). Um argumento análogo mostra que a classe **coRP** está contida na classe **coNP**.

Também é claro que **P** está contida nessas duas classes, pois uma MTD é um caso especial de uma máquina de Turing Monte-Carlo, onde as palavras sempre são aceitas e rejeitadas corretamente.

## As classes ZPP e BPP

Na última seção, vimos classes de linguagens que são decididas por MTPs limitadas polinomialmente que podem ou aceitar ou rejeitar erroneamente uma determinada palavra.

Agora, vamos definir uma classe de linguagens que exige das máquinas que as respostas sejam corretas, mas que abre mão da certeza da polinomialidade na execução.

Para que uma linguagem $L$ pertença à classe **ZPP** (*zero-error probability polynomial-time*), deve existir uma máquina de Turing probabilística $M$ que sempre aceita palavras que pertencem a $L$ e sempre rejeita palavras que não pertencem a $L$, e cujo tempo esperado de execução para qualquer entrada é polinomial.

Pela definição, podemos perceber que a classe **ZPP** é muito parecida com a classe **P**, diferindo apenas no fato de que as máquinas utilizadas podem consumir tempo superpolinomial em algumas computações.

Outra definição possível é a seguinte: **ZPP** é a classe das linguagens que pertencem tanto a **RP** como a **coRP**. Se uma linguagem está tanto em **RP** como em **coRP**, executa-se de maneira alternada cada uma das máquinas envolvidas até que uma delas obtenha uma resposta certamente correta. O número de passos desse algoritmo é indeterminado (não há sequer garantias de que ele vai parar em algum momento), mas a esperança do número de passos é um valor polinomial no tamanho da entrada, pois a probabilidade de obtenção de respostas erradas diminui exponencialmente a cada execução das máquinas.

Outra classe de complexidade importante, que também pode ser relacionada com as demais classes probabilísticas já citadas, é a classe **BPP** (*bounded-error probabilistic polynomial-time*), que contém as linguagens para as quais existem máquinas de Turing que devolvem respostas erradas (tanto rejeições como aceitações) com probabilidade estritamente menor que 0,5. Na verdade, qualquer delimitação da probabilidade no intervalo $(0; 0{,}5)$ resultaria na mesma classe. Por outro lado, Papadimitriou [Pap94] mostrou que delimitações maiores ou iguais a 0,5 podem levar a uma classe diferente.

A definição dessa classe é simétrica, pois rejeição e aceitação devem ser feitas corretamente na maioria absoluta das computações feitas por essas máquinas. Assim, utilizando um argumento análogo ao do resultado $\mathbf{P} = \mathbf{coP}$, obtemos que $\mathbf{BPP} = \mathbf{coBPP}$.

Além disso, é fácil ver que $\mathbf{RP} \subseteq \mathbf{BPP}$. Dadas uma linguagem $L$ em **RP** e uma máquina $M$ associada, basta executar $M$ com a entrada desejada duas vezes para que a probabilidade de falsa rejeição seja de no máximo 0,25. Como a probabilidade de falsa aceitação é zero, as condições necessárias para que $L$ pertença a **BPP** estão satisfeitas. O argumento que mostra que $\mathbf{coRP} \subseteq \mathbf{BPP}$ é análogo.

Por fim, vale destacar que não sabemos ainda se a classe **BPP** está contida na classe **NP**.

## 6.2 Classes quânticas de complexidade

Vamos agora apresentar as duas principais classes quânticas de complexidade e demonstrar alguns resultados que as relacionam com as classes clássicas.

## 6.3 Recursos e linguagens no modelo quântico

Durante as discussões sobre o modelo clássico de computação, fizemos considerações sobre o consumo de tempo e de espaço pelas máquinas de Turing determinísticas, não-determinísticas e probabilísticas.

O tempo consumido por uma MTQ com entrada $x$ é o número de passos que a máquina efetua até terminar a execução. Essa definição é consistente, porque exigimos que, quando uma configuração da superposição atinge um estado final, todas as demais configurações também o fazem.

O espaço consumido por uma MTQ $M$ com entrada $x$ é definido de maneira análoga ao espaço consumido por uma MTND. Dentre todas as configurações que durante a execução de $M$ tiveram amplitude não-nula, seja $c$ aquela com o maior número possível de células em que $M$ escreveu algo. O número de células utilizadas em $c$ é o espaço consumido por $M$ com a entrada $x$.

No contexto de linguagens, estamos interessados em MTQs que, para cada entrada $x$, têm saída ou $x1$ ou $x0$ (MTQs devem ser reversíveis, por isso a resposta usualmente binária é precedida pela entrada). Diz-se que uma MTQ $M$ desse tipo decide de maneira exata uma linguagem $L$ se, para todo $x$ em $L$, a máquina $M$ produz como saída $x1$, e, para todo $x$ fora de $L$, a máquina $M$ tem como saída $x0$.

Por fim, para um número $p$ entre 0 e 1, diz-se que uma MTQ $M$ decide $L$ com probabilidade $p$ se $M$, com entrada $x$ em $L$, produz $x1$ com probabilidade pelo menos $p$ e, com entrada $x$ fora de $L$, produz $x0$ com probabilidade pelo menos $p$.

## 6.4 As classes EQP e BQP

As duas classes de complexidade que iremos introduzir aqui foram propostas por Bernstein e Vazirani [BV97].

A classe **EQP** é o conjunto das linguagens que são decididas de maneira exata por uma MTQ que consome tempo polinomial no tamanho da entrada. Essa classe corresponde à classe **P** do modelo clássico.

A classe **BQP** é o conjunto das linguagens para as quais existem máquinas de Turing quânticas que devolvem respostas erradas (tanto rejeições como aceitações) com probabilidade estritamente menor que 0,5. A classe correspondente no modelo clássico é **BPP**, e assim como no caso dela, a probabilidade de erro pode ser qualquer valor no intervalo $(0; 0{,}5)$.

A seguir, vamos mostrar algumas relações envolvendo essas classes e as classes tradicionais de complexidade.

## 6.5 Relações envolvendo as classes quânticas

Inicialmente, vamos mostrar que $\mathbf{P} \subseteq \mathbf{EQP}$ e que $\mathbf{BPP} \subseteq \mathbf{BQP}$, pois as demonstrações e as idéias envolvidas são mais simples. Em seguida, mostraremos que $\mathbf{BQP} \subseteq \mathbf{PSPACE}$. Seguem abaixo demonstrações breves das duas primeiras relações, apresentadas no artigo de Bernstein e Vazirani [BV97].

**Teorema 6.1.** $\mathbf{P} \subseteq \mathbf{EQP}$.

*Demonstração.* Seja $L$ uma linguagem que pertence à classe $\mathbf{P}$. É fácil ver que existe uma MTD limitada polinomialmente que, para cada entrada $x$, produz como saída $x1$ se $x \in L$ e $x0$ caso contrário. Para toda MTD polinomialmente limitada, existe uma MTD reversível equivalente, que também é limitada polinomialmente. Para obtê-la, modifica-se a máquina original de modo que cada um de seus estados só possa ser atingido por movimentos da cabeça de leitura em uma única direção (as transições cujas triplas têm o mesmo estado devem fazer o mesmo deslocamento da cabeça de leitura da máquina).

Isso pode ser feito por meio da duplicação dos estados da máquina. Desse modo, a função de transição torna-se bijetora quando a direção é ignorada, e a partir de uma configuração arbitrária da máquina, é possível deduzir a configuração "anterior". Ou seja, a máquina é reversível. Além disso, o número de passos executados pela máquina até que ela pare é o mesmo da máquina original.

Mas se uma MTD é reversível, então essa máquina é uma MTQ onde cada superposição de configurações contém apenas uma configuração com amplitude não-nula.

Logo, existe uma MTQ limitada polinomialmente que, para cada entrada $x$, devolve $x1$ como resposta sempre que $x$ está em $L$, e devolve $x0$ sempre que $x$ está fora de $L$. Então, $L$ é decidida de maneira exata por uma MTQ, e portanto pertence a $\mathbf{EQP}$. Concluímos assim que $\mathbf{P} \subseteq \mathbf{EQP}$. □

A demonstração a seguir mostra como uma MTQ engloba naturalmente um gerador de números aleatórios.

**Teorema 6.2.** $\mathbf{BPP} \subseteq \mathbf{BQP}$.

*Demonstração.* Sejam $L$ uma linguagem em $\mathbf{BPP}$ e $M$ uma MTP para $L$ dada pela definição de $\mathbf{BPP}$. Para mostrar que $L \in \mathbf{BQP}$, vamos descrever uma MTQ que simula $M$ com um aumento apenas polinomial no consumo de tempo.

Para simplificar, vamos assumir que $M$ tem $k$ opções de transição em cada um de seus passos, para algum $k$ fixo. Se numa simulação de $M$ por uma MTQ $M'$ conseguimos, a cada passo, escolher aleatoriamente um símbolo do alfabeto

$\{1, \ldots, k\}$, então $M$ pode ser simulada por $M'$. Cada símbolo do alfabeto é usado para definir qual transição se aplica no passo que está sendo simulado de $M$. Esses símbolos podem ser gerados da seguinte maneira.

A cada passo, se o estado atual não é final, a cabeça de leitura desloca-se para uma determinada posição da fita e escreve-se o símbolo $j$ com amplitude $1/\sqrt{k}$, para $j = 1, \ldots, k$, em uma única transição, que representa um sorteio aleatório. Feito isso, o conteúdo da célula é lido, a cabeça retorna para a posição original da fita e a transição representada pelo símbolo lido é realizada.

A célula utilizada no sorteio deve ser tal que não haja interferência no resto da fita. Como $M$ é polinomialmente limitada, existem números $c$ e $e$ tais que qualquer computação de $M$ não consome mais do que $cn^e$ passos. Assim, basta inserir os caracteres dos sorteios seqüencialmente, começando na $(cn^{e+1})$-ésima célula da fita.

Como cada transição do passo de sorteio tem amplitude $1/\sqrt{k}$, então cada configuração no final terá amplitude $(1/\sqrt{k})^t$, onde $t$ é o número de passos de $M$, lembrando que não ocorrerá interferência entre as configurações geradas. Logo, com a medição, a probabilidade de obtenção de um determinado resultado em $M'$ será igual à da obtenção do mesmo resultado em $M$.

Como $M$ pode ser simulada por $M'$, a linguagem $L$ pode ser decidida com a mesma probabilidade $p > 0{,}5$ por uma MTQ limitada polinomialmente. Logo, $L \in \mathbf{BQP}$, e portanto temos que $\mathbf{BPP} \subseteq \mathbf{BQP}$. □

As duas provas mostradas acima servem para reforçar idéias que devem ser intuitivas após a leitura das definições envolvidas. Nosso objetivo agora é mostrar uma relação menos imediata e mais importante, que envolve o consumo de espaço na simulação de uma MTQ por uma MTD.

Quando falamos em simulação de uma MTQ por uma MTD, estamos nos referindo a uma simulação onde queremos saber a probabilidade de cada saída possível ser produzida. Não conhecemos nenhuma simulação de uma MTQ por uma MTD que consuma tempo polinomial no tempo consumido pela MTQ, porém o teorema abaixo mostra que é possível fazer uma simulação utilizando espaço polinomial no espaço consumido pela MTQ. Nesse texto, mostraremos apenas as linhas gerais da prova desse teorema.

**Teorema 6.3.** $\mathbf{BQP} \subseteq \mathbf{PSPACE}$.

*Demonstração.* Seja $L \in \mathbf{BQP}$ e $M := (Q, \Sigma, \alpha, s, H)$ uma MTQ polinomialmente limitada que decide $L$ com probabilidade $p$ maior que 0,5. Vamos descrever uma MTD $M'$ que decide $L$ em espaço polinomial no tempo consumido por $M$. A máquina $M'$ calcula, para cada entrada $x$, a probabilidade $p_x$

de $M$ aceitar $x$ (ou seja, de $M$ produzir $x1$ como saída). No final, $M'$ aceita ou rejeita $x$ de acordo com o valor de $p_x$.

Dizemos que uma configuração $(w_1, q, w_2)$ tem *tamanho* $k$ se a cadeia de caracteres $w_1w_2$ tem $k$ caracteres. Dadas duas configurações $c_1$ e $c_2$, denotamos por $\alpha(c_1, c_2)$ o valor da amplitude correspondente à transição de $M$ que leva $c_1$ a $c_2$. Se não há nenhuma transição que leva $c_1$ a $c_2$, então $\alpha(c_1, c_2) = 0$. Seja $c_0 := (\rhd, s, x)$ a configuração inicial.

A máquina $M'$, para cada inteiro $t$ a partir de $|x|$, simula $t$ passos de $M$ e calcula a probabilidade $p_x$ de $M$ produzir, em $t$ passos, a saída $x1$. Essa probabilidade é dada por

$$p_x = \left| \sum_{c_1,\ldots,c_t} \prod_i \alpha(c_{i-1}, c_i) \right|^2,$$

onde o somatório é sobre todas as configurações $c_1, \ldots, c_t$ de tamanho no máximo $t$ onde $c_t = (\rhd, h, x_1)$, para algum $h$ em $H$. Note que o número de termos no somatório é no máximo $T = t|\Sigma|^{2t}|Q|$.

A máquina $M'$ deve portanto calcular o somatório descrito acima. O primeiro problema que aparece é a incapacidade de $M'$ fazer cálculos com números irracionais. MTDs só podem armazenar valores inteiros limitados, enquanto que cada $\alpha(c_{i-1}, c_i)$ pode nem mesmo ser racional. Logo, a simulação será uma aproximação, que deverá ter um erro tolerável. Por fim, vale destacar que cada um dos no máximo $T$ termos do somatório terá $t$ fatores.

Se cada $\alpha(c_{i-1}, c_i)$ for representado com $m$ bits tanto na parte inteira como na parte imaginária dos números, onde $m$ é grande comparado a $\log T$, o erro será pequeno (da ordem de $2^{-m}$). Assim, a primeira dificuldade já foi eliminada.

O que $M'$ deve fazer é computar cada termo do somatório e guardar a soma dos termos. Como cada operação de adição usa pouco espaço auxiliar (ou seja, consome espaço polinomial em $m$) e só é necessário guardar a cada operação a soma atual, podemos garantir que o espaço necessário para efetuar o somatório é polinomial. No caso dos produtos, os mesmos comentários se aplicam.

O que resta considerar agora é a obtenção de cada $\alpha(c_{i-1}, c_i)$. É fácil, em tempo polinomial nos comprimentos de $c_{i-1}$ e $c_i$, detectar se $c_i$ pode ou não ser derivada de $c_{i-1}$ em um passo. Se não pode, então $\alpha(c_{i-1}, c_i) = 0$. Se pode, então ao mesmo tempo pode-se determinar a transição $(q_1, \sigma_1, q, \sigma, d)$ em $Q \times \Sigma \times Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$ que, quando aplicada a $c_{i-1}$, leva a $c_i$. A máquina $M'$ então aciona uma "subrotina" que recebe um número $m$ e devolve, em espaço polinomial em $m$, o valor de $\alpha(q_1, \sigma_1, q, \sigma, d)$ com precisão $2^{-m}$. O valor devolvido pela subrotina é a aproximação desejada de $\alpha(c_{i-1}, c_i)$. Mostrar que de fato há uma MTD que efetua tal subrotina e consome espaço polinomial em $m$ não é tarefa trivial e envolve uma série de etapas. Omitiremos essa prova nesse texto.

É importante notar que o número de tais subrotinas não depende da entrada, mas apenas da máquina $M$. Assim $M'$ está bem definida, desde que existam MTDs que executem tais subrotinas. Mais do que isso, cada etapa que $M'$ executa consome espaço polinomial (assumindo que as subrotinas consumam espaço polinomial), portanto de fato $\mathbf{BQP} \subseteq \mathbf{PSPACE}$. $\square$

Dos resultados acima, temos que $\mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{PSPACE}$. Assim, não é possível mostrar que $\mathbf{BPP} \neq \mathbf{BQP}$ sem resolver a clássica questão de se $\mathbf{P}$ está propriamente contido em $\mathbf{PSPACE}$ ou não. Por outro lado, no mesmo artigo, Bernstein e Vazirani [BV97] mostram um oráculo em relação ao qual $\mathbf{BPP} \neq \mathbf{BQP}$. Outros resultados nessa linha, porém direcionados à questão "$\mathbf{P} \neq \mathbf{NP}$?", foram provados por Bennet *et al.* [BBBV97]. Por exemplo, Bennet *et al.* mostraram que, em relação a um oráculo, $\mathbf{NP} \not\subseteq \mathbf{BQP}$.

# Parte III

# Apêndices

# Apêndice A

# Espaços de Hilbert

Os espaços de Hilbert fornecem o formalismo apropriado para o estudo de conceitos da mecânica quântica e, portanto, da computação quântica.

Neste capítulo definimos espaços de Hilbert, estabelecemos nossas notações e listamos algumas propriedades importantes.

## A.1 Espaços vetoriais, produto interno e norma

**Definição A.1.** *Um* espaço vetorial $\mathcal{S}$, *com um conjunto* $H$ *sobre* $\mathcal{K}$, *é uma álgebra* $\mathcal{S} = \langle H, +, ^{-1}, \mathbf{0}, K, +_f, \times_f, 0, 1, \cdot \rangle$ *tal que* $\langle H, +, ^{-1}, \mathbf{0} \rangle$ *é um grupo comutativo,* $\mathcal{K} = \langle K, +_f, \times_f, 0, 1 \rangle$ *é um corpo, e* $\cdot : K \times H \longrightarrow H$ *é uma multiplicação escalar satisfazendo os seguintes axiomas para quaisquer* $a, b \in K$ *e quaisquer* $\phi, \psi \in H$*:*

- $a \cdot (\phi + \psi) = a \cdot \phi + a \cdot \psi, \quad (a +_f b) \cdot \phi = a \cdot \phi + b \cdot \phi$
- $\big(a \cdot (b \cdot \phi)\big) = (a \times_f b) \cdot \phi$
- $1 \cdot \phi = \phi$.

Feita esta definição, a partir de agora não faremos mais distinção entre os operadores $+$ e $+_f$, $\cdot$ e $\times_f$, exceto quando necessário. Além disso, escreveremos $\phi - \psi$ no lugar de $\phi + \psi^{-1}$.

**Definição A.2.** *Um* espaço com produto interno complexo $H$ *é um espaço vetorial com um conjunto* $H$ *sobre o corpo dos números complexos, munido de um produto interno* $\langle \cdot | \cdot \rangle : H \times H \longrightarrow \mathbb{C}$ *satisfazendo os seguintes axiomas. Para quaisquer* $\phi, \phi', \psi \in H$ *e quaisquer* $a, b \in \mathbb{C}$*:*

- $\langle \psi | \phi \rangle = \langle \phi | \psi \rangle^*$

- $\langle\psi|\psi\rangle \geq 0$, *com* $\langle\psi|\psi\rangle = 0$ *se, e somente se,* $\psi = 0$
- $\langle\psi|a\phi + b\phi'\rangle = a\langle\psi|\phi\rangle + b\langle\psi|\phi'\rangle$.

**Definição A.3.** *Sejam* $\mathcal{S}$ *um espaço com produto interno complexo e* $\phi, \psi \in \mathcal{S}$. *A* norma *de* $\phi$ *é definida por* $\|\phi\| := \sqrt{\langle\phi|\phi\rangle}$. *A* distância *entre os vetores* $\phi$ *e* $\psi$ *é dada por* $\operatorname{dist}(\phi, \psi) := \|\phi - \psi\|$.

Se tivermos $H = \mathbb{C}^n$ e o produto interno definido como

$$\big\langle(x_1, \ldots, x_n)\big|(y_1, \ldots, y_n)\big\rangle = \sum_{i=1}^{n} x_i^* y_i,$$

então falamos sobre o *espaço com produto interno complexo n-dimensional.*

Neste texto adotamos a notação de Dirac, de modo que nos referimos a um elemento $\psi$ de um espaço com produto interno complexo por $|\psi\rangle$, que é chamado um vetor *ket*. Como pode ser visto no restante do texto e deste capítulos, tal notação é bastante conveniente.

Considere um espaço com produto interno complexo $H$. Para quaisquer vetores $\phi, \psi \in H$ e qualquer $c \in \mathbb{C}$, valem as seguintes propriedades:

(a) $\langle c\phi|\psi\rangle = c^*\langle\phi|\psi\rangle$

(b) $\|c\phi\| = |c|\,\|\phi\|$

(c) $|\langle\phi|\psi\rangle| \leq \|\phi\|\|\psi\|$ (desigualdade de Cauchy-Schwarz)

(d) $\|\phi + \psi\| \leq \|\phi\| + \|\psi\|$ (desigualdade triangular)

(e) $\|\phi + \psi\|^2 + \|\phi - \psi\|^2 = 2\|\phi\|^2 + 2\|\psi\|^2$ (lei do paralelogramo)

*Demonstração.* (a) Utilizando os axiomas do produto interno, temos:

$$\langle c\phi|\psi\rangle = \langle\psi|c\phi\rangle^* = \big(c\langle\psi|\phi\rangle\big)^* = c^*\langle\psi|\phi\rangle^* = c^*\langle\phi|\psi\rangle.$$

(b) Utilizando a propriedade (a) provada acima, temos:

$$\|c\phi\| = \sqrt{\langle c\phi|c\phi\rangle} = \sqrt{c^*c\langle\phi|\phi\rangle} = \sqrt{c^*c}\,\sqrt{\langle\phi|\phi\rangle} = |c|\,\|\phi\|.$$

(c) Caso $\langle\phi|\psi\rangle = 0$, a desigualdade se reduz a $0 \leq \|\phi\|\|\psi\|$ e é trivialmente satisfeita. Senão, $\langle\phi|\psi\rangle \neq 0$ e podemos assumir, sem perda de generalidade, que $\psi \neq 0$. Suponha então que $\psi \neq 0$ e $\langle\phi|\psi\rangle \neq 0$.

Para qualquer $c \in \mathbb{C}$, temos $\langle \phi + c\psi | \phi + c\psi \rangle \geq 0$. Expandindo essa expressão, temos $\langle \phi|\phi \rangle + (c\langle \phi|\psi \rangle)^* + c\langle \phi|\psi \rangle + c^* c\langle \psi|\psi \rangle \geq 0$. Em particular, tome

$$c = \lambda \frac{\langle \phi|\psi \rangle^*}{|\langle \phi|\psi \rangle|}, \text{ com } \lambda = \lambda^*.$$

Então teremos $\lambda^2 \langle \psi|\psi \rangle + 2\lambda |\langle \phi|\psi \rangle| + \langle \phi|\phi \rangle \geq 0$.

Como $\psi \neq 0$, $g(\lambda) = \lambda^2 \langle \psi|\psi \rangle + 2\lambda |\langle \phi|\psi \rangle| + \langle \phi|\phi \rangle$ é um trinômio do segundo grau em $\lambda$. Além disso, sabemos que $g(\lambda) \geq 0$, e portanto seu discriminante deve ser não-positivo. Assim, $|\langle \phi|\psi \rangle| - \langle \phi|\phi \rangle \langle \psi|\psi \rangle \leq 0$, e a desigualdade de Cauchy-Schwarz segue imediatamente.

(d) O resultado segue da desigualdade de Cauchy-Schwarz. Abaixo denotamos a parte real de um número complexo $c$ por $\mathrm{Re}(c)$:

$$\begin{aligned}
\|\phi + \psi\|^2 &= \langle \phi + \psi | \phi + \psi \rangle \\
&= \langle \phi|\phi \rangle + \langle \phi|\psi \rangle + \langle \psi|\phi \rangle + \langle \psi|\psi \rangle \\
&= \langle \phi|\phi \rangle + 2\,\mathrm{Re}(\langle \phi|\psi \rangle) + \langle \psi|\psi \rangle \\
&\leq \langle \phi|\phi \rangle + 2\big|\mathrm{Re}(\langle \phi|\psi \rangle)\big| + \langle \psi|\psi \rangle \\
&\leq \|\phi\|^2 + 2\|\phi\|\|\psi\| + \|\psi\|^2, \text{por (c)} \\
&= \big(\|\phi\| + \|\psi\|\big)^2.
\end{aligned}$$

(e) Segue diretamente da expansão da norma, de acordo com a definição.

□

**Definição A.4.** *Dada uma seqüência infinita $\phi_1, \phi_2, \ldots \in H$, onde $H$ é um espaço com produto interno complexo, dizemos que ela* converge *para $\phi \in H$ se, para qualquer $\epsilon > 0$, existir um número $N(\epsilon)$ tal que $\|\phi_i - \phi\| < \epsilon$ para todo $i > N(\epsilon)$.*

*Além disso, tal seqüência infinita é chamada de* seqüência de Cauchy *se, para qualquer $\epsilon > 0$, existir um número $M(\epsilon)$ tal que $\|\phi_i - \phi_j\| < \epsilon$ para quaisquer $i, j > M(\epsilon)$.*

Se toda seqüência de Cauchy em $H$ converge, dizemos que $H$ é completo.

**Definição A.5.** *Um* espaço de Hilbert *é um espaço completo com produto interno complexo.*

Não vamos nos deter nos detalhes topológicos, que são desnecessários aqui. Também não vamos definir independência linear, bases, conjuntos geradores e dimensão, pois esses conceitos já devem ser bem conhecidos.

A partir de agora vamos nos referir sempre a $H$ como um espaço de Hilbert.

## A.2 Espaço de Hilbert conjugado

**Definição A.6.** *Uma função* $f : H \longrightarrow \mathbb{C}$ *é dita* linear *se* $f(a\phi + b\psi) = af(\phi) + bf(\psi)$*, para quaisquer* $\phi, \psi \in H$ *e quaisquer* $a, b \in \mathbb{C}$*. Dizemos também que* $f$ *é um* funcional.

É possível provar que para todo funcional $f$ existe um único $\phi_f \in H$ tal que $f(\psi) = \langle \phi_f | \psi \rangle$ para todo $\psi \in H$.

O conjunto de todos os funcionais de um espaço de Hilbert $H$ é também um espaço de Hilbert (se definirmos a adição e a multiplicação da maneira usual). Tal espaço é denominado *dual do espaço de Hilbert* $H$ ou *espaço de Hilbert conjugado*, denotado por $H^*$, com produto interno $\langle f | g \rangle = \langle \phi_f | \phi_g \rangle$ para todo $f, g \in H^*$.

Desta forma, estabelece-se uma bijeção entre $H$ e $H^*$ e portanto todo espaço de Hilbert é isomorfo ao seu dual.

Dado um elemento $\phi \in H$, ou $|\phi\rangle$ como vetor *ket* na notação de Dirac, o funcional correspondente é denotado por $\langle \phi |$, denominado vetor *bra*. Para qualquer $\psi \in H$, temos $\langle \phi | \big( |\psi\rangle \big) = \langle \phi | \psi \rangle$.

Se estivermos falando de um espaço de Hilbert $n$-dimensional, podemos pensar em $|\phi\rangle$ como um vetor coluna de dimensão $n$ e $\langle \phi |$ como um vetor linha de mesma dimensão. Neste caso, a transformação $|\phi\rangle \leftrightarrow \langle \phi |$ corresponde a uma transposição e conjugação. O produto externo $|\phi\rangle\langle\psi|$ equivale portanto a uma matriz de dimensão $n$, utilizando-se a definição usual de multiplicação de matrizes. O mesmo se aplica ao cálculo do produto interno.

## A.3 Bases ortonormais

A ortogonalidade é um conceito de extrema importância para a computação quântica. Numa medição, só é possível distingüir estados quânticos mutuamente ortogonais.

**Definição A.7.** *Dois vetores não-nulos* $\phi, \psi \in H$ *são* ortogonais*, o que se denota por* $\phi \perp \psi$*, se* $\langle \phi | \psi \rangle = 0$*. Um conjunto* $S \subseteq H$ *é* ortogonal *se cada par de elementos distintos do conjunto forem ortogonais. Um conjunto ortogonal é chamado* ortonormal *se todos seus elementos têm norma* 1.

**Proposição A.8.** *Seja* $B \subseteq H$ *um conjunto ortogonal com* $n$ *elementos. Então os vetores de* $B$ *são linearmente independentes.*

*Demonstração.* Seja $B = \{\phi_1, \ldots, \phi_n\}$. Vamos provar que a única solução para $a_1\phi_1 + \ldots + a_n\phi_n = 0$ é a trivial, com $a_i = 0$ para todo $1 \leq i \leq n$, e portanto os vetores $\phi_1, \ldots, \phi_n$ são linearmente independentes.

Dado $i$, temos que $\langle\phi_i|a_1\phi_1 + \cdots + a_n\phi_n\rangle = \langle\phi_i|\ 0\rangle = 0$. Porém, temos também que $\langle\phi_i|a_1\phi_1+\cdots+a_n\phi_n\rangle = \sum_{j=1}^{n} a_j\langle\phi_i|\phi_j\rangle = 0$. Como $B$ é ortogonal, $\langle\phi_i|\phi_j\rangle = 0$ se $i \neq j$ e portanto a soma anterior é simplificada para $a_i\langle\phi_i|\phi_i\rangle = 0$. Como $\langle\phi_i|\phi_i\rangle \neq 0$, devemos ter $a_i = 0$.

Como $i$ foi escolhido arbitrariamente, temos que $a_i = 0$ para $1 \leq i \leq n$ e portanto os vetores de $B$ são linearmente independentes. □

**Definição A.9.** *Um conjunto ortonormal $B \subseteq H$ é uma* base ortonormal *de $H$ se todo vetor $v \in H$ pode ser escrito como*

$$v = \sum_{\phi\in B} a_\phi \phi,$$

*com $a_\phi \in \mathbb{C}$ para todo $\phi \in B$.*

**Definição A.10.** *Seja $B = \{\phi_i\}_{i=1}^{n}$ uma base para um espaço de Hilbert $H$ e $\psi \in H$ um vetor. Então $\psi$ pode ser escrito como combinação linear dos elementos de $B$: $\psi = a_1\phi_1+\cdots+a_n\phi_n$. Chamamos $(a_1, \ldots, a_n)$ a* representação *de $\psi$ na base $B$.*

Pode-se provar que quaisquer espaços de Hilbert $H_1$ e $H_2$ de mesma dimensão são isomorfos. Assim, denotaremos um espaço de Hilbert $d$-dimensional por $H_d$.

Dada uma base ortonormal $B$ de um espaço de Hilbert $H$, é fácil verificar as seguintes propriedades, para quaisquer $\phi, \psi \in H$:

1. $|\phi\rangle = \sum_{\gamma\in B}\langle\gamma|\phi\rangle|\gamma\rangle$
2. Se $\langle\phi|\gamma\rangle = 0$ para todo $\gamma \in B$, então $\phi = 0$.
3. $\langle\phi|\psi\rangle = \sum_{\gamma\in B}\langle\phi|\gamma\rangle\langle\gamma|\psi\rangle$
4. $\|\psi\|^2 = \sum_{\gamma\in B}\left|\langle\gamma|\psi\rangle\right|^2$ (identidade de Parseval)

Vamos provar apenas o primeiro item. Todos os outros seguem imediatamente deste.

*Demonstração.* Como $B = \{\gamma_1, \ldots, \gamma_n\}$ é uma base, existem $a_1, \ldots, a_n \in \mathbb{C}$ tais que $|\phi\rangle = \sum_{i=1}^{n} a_i|\gamma_i\rangle$.

Dado $i$, temos que

$$\langle\gamma_i|\phi\rangle = \sum_{j=1}^{n} a_i\langle\gamma_i|\gamma_j\rangle = a_i\langle\gamma_i|\gamma_i\rangle = a_i \cdot 1 = a_i.$$

□

Dado um subespaço $W$ de um espaço de Hilbert $H$, existe um único subespaço $W^\perp$ de $H$ tal que todo vetor de $W$ é ortogonal a todo vetor de $W^\perp$. Além disso, todo vetor $\psi \in H$ pode ser escrito de forma única como $\psi = \phi_1 + \phi_2$, com $\phi_1 \in W$ e $\phi_2 \in W^\perp$.

Nesse caso dizemos que $H$ é a *soma direta* de $W$ e $W^\perp$ e denotamos por $H = W \oplus W^\perp$. Dizemos também que $W$ e $W^\perp$ formam uma *decomposição ortogonal* de $H$.

Podemos generalizar essa decomposição em subespaços ortogonais. Assim, $H = W_1 \oplus \cdots \oplus W_n$, onde $\langle \phi_i | \phi_j \rangle = 0$ sempre que $\phi_i \in W_i$ e $\phi_j \in W_j$, com $i \neq j$, onde $W_i, 1 \leq i \leq n$, são subespaços de $H$. Segue também que todo $\psi \in H$ pode ser escrito de forma única como $\psi = \phi_1 + \cdots + \phi_n$ com $\phi_i \in W_i, 1 \leq i \leq n$.

## A.4 Operadores lineares

**Definição A.11.** *Um* operador linear *sobre um espaço de Hilbert $H$ é uma função linear $A : H \longrightarrow H$.*

A aplicação de um operador linear $A$ em um vetor $|\psi\rangle$ é denotada por $A|\psi\rangle$.

Além disso, $A$ também é um operador linear de $H^*$, levando $\langle \phi|$ a $\langle \phi|A$. Quando $\langle \phi|A$ é aplicado a algum vetor $\psi$, primeiro realiza-se a aplicação de $A$ sobre $\psi$ para depois aplicar $\langle \phi|$.

Um operador linear $A$ é chamado *positivo* ou *semi-definido*, denotado por $A \geq 0$, se, para qualquer $|\psi\rangle \in H$, tivermos $\langle \psi | A\psi \rangle \geq 0$.

Fixada uma base enumerável $B = \big\{ |\theta_i\rangle : i \in I \big\}$ de $H$, qualquer operador linear $A$ pode ser representado por uma matriz, indexada por elementos de $I$, tendo, como entrada $(i, j)$, o valor $\langle \theta_i | A\theta_j \rangle$. Essa matriz é identificada com o próprio operador $A$ e denotada da mesma forma, quando não houver perigo de confusões por conta de bases diferentes.

**Definição A.12.** *A* norma $\|A\|$ *de um operador linear $A$ é* $\sup_{\|\phi\|=1} \|A\phi\|$. *Um operador linear $A$ é dito* limitado *se $\|A\| < \infty$.*

Uma classe de operadores lineares de extrema importância são os *projetores*. Como foi visto anteriormente, se tivermos uma decomposição ortogonal de $H$ em $W_1$ e $W_2$, então todo vetor de $\psi \in H$ tem uma representação única $\psi = \psi_1 + \psi_2$, com $\psi_i \in W_i, i = 1, 2$. Os vetores $\psi_1$ e $\psi_2$ são as *projeções* de $\psi$ nos subespaços $W_1$ e $W_2$, respectivamente. Neste caso, o operador $P_{W_i}$ que leva $\psi$ a $\psi_i$ é chamado *projetor sobre o subespaço $W_i$*.

Se um subespaço é gerado apenas por um vetor $\phi$, $\|\phi\| = 1$, escrevemos $P_\phi$. Pode-se provar que $P_\phi = |\phi\rangle\langle\phi|$. Veja [TI97].

Também não é difícil provar que todo projetor é *idempotente*, ou seja, que $P^2 = P$.

**Definição A.13.** *O* adjunto *$T^*$ de um operador linear $T$ limitado é um operador tal que, para quaisquer $\phi, \psi \in H$, $\langle \psi | T\phi \rangle = \langle T^*\psi | \phi \rangle$. Um operador $T$ tal que $T^* = T$ é denominado* auto-adjunto.

Normalmente utilizaremos a notação $\langle \psi | T | \phi \rangle$, ao invés de $\langle \psi | T\phi \rangle$, de modo que

$$\langle T^*\psi | \phi \rangle = \langle \psi | T | \phi \rangle = \langle \psi | T\phi \rangle.$$

Se tomarmos a matriz $A$ correspondente ao operador linear $T$ em relação a uma base qualquer, a matriz correspondente a $T^*$ é simplesmente $A^*$, a conjugada transposta de $A$.

Uma matriz $A$ é *hermitiana* se $A = A^*$. A cada operador auto-adjunto $T$ corresponde uma matriz hermitiana $A$.

## A.5 Autovalores, autovetores e representação espectral

**Definição A.14.** *Dado um operador $A$ sobre o espaço de Hilbert $H_n$, $\lambda \in \mathbb{C}$ e $\phi \in H_n$, se $A\phi = \lambda\phi$, então $\phi$ é um* autovetor *de $A$ e $\lambda$ é o* autovalor associado *a $\phi$. O* espectro *de $A$ é o conjunto de todos seus autovalores.*

Para um dado operador $A$, se tivermos $A\phi = \lambda\phi$, $\lambda \in \mathbb{C}$, então $A(c\phi) = c(A\phi) = c\lambda\phi = \lambda(c\phi)$ pela linearidade de $A$. Logo $c\phi$ também é um autovetor de $A$. Se $A\phi_1 = \lambda\phi_1$ e $A\phi_2 = \lambda\phi_2$, com $\phi_1 \neq \phi_2$, segue da mesma forma que $A(\phi_1 + \phi_2) = \lambda(\phi_1 + \phi_2)$, de modo que $\phi_1 + \phi_2$ também é autovetor de $A$. Como $A \cdot 0 = \lambda \cdot 0$, temos que o conjunto de todos os autovetores associados a $\lambda$ é um subespaço de $H$, chamado de *autoespaço* de $H$ e denotado por $H_\lambda$.

Se os autovalores distintos de uma matriz hermitiana $A$ são $\lambda_1, \ldots, \lambda_m$ e a dimensão do autoespaço $H_{\lambda_i}$ é $m(i)$, então $\sum_{i=1}^{m} m(i) = n$, ou seja, o subespaço gerado por todos os autovetores tem dimensão $n$.

**Proposição A.15.** *O determinante de uma matriz é igual ao produto de seus autovalores. Além disso, o traço (soma dos elementos da diagonal) de uma matriz $A$, denotado por* $\mathrm{Tr}(A)$, *é igual à soma dos autovalores de $A$.*

A prova deste resultado pode ser vista em [TI97].

**Proposição A.16.** *Todos os autovalores de uma matriz hermitiana $A$ são reais. Além disso, autovetores de $A$ correspondendo a autovalores diferentes são ortogonais.*

*Demonstração.* Seja $\phi$ um autovetor de $A$ correspondendo ao autovalor $\lambda$. Temos então

$$\lambda\langle\phi|\phi\rangle = \langle\phi|\lambda\phi\rangle = \langle\phi|A\phi\rangle = \langle A^*\phi|\phi\rangle = \langle A\phi|\phi\rangle = \langle\lambda\phi|\phi\rangle = \lambda^*\langle\phi|\phi\rangle.$$

Como $\phi \neq 0$, temos $\lambda = \lambda^*$ e assim $\lambda \in \mathbb{R}$.

Considere agora $\phi_1, \phi_2$ dois autovetores de $A$, correspondentes aos autovalores $\lambda_1, \lambda_2 \in \mathbb{R}$, respectivamente, com $\lambda_1 \neq \lambda_2$. Temos então

$$\begin{aligned} \lambda_1\langle\phi_1|\phi_2\rangle &= \langle\lambda_1\phi_1|\phi_2\rangle = \langle A\phi_1|\phi_2\rangle = \langle A^*\phi_1|\phi_2\rangle \\ &= \langle\phi_1|A\phi_2\rangle = \langle\phi_1|\lambda_2\phi_2\rangle = \lambda_2\langle\phi_1|\phi_2\rangle. \end{aligned}$$

Como $\lambda_1 \neq \lambda_2$, teremos necessariamente $\phi_1 \perp \phi_2$. □

Considere um espaço de Hilbert $H_n$ e uma matriz hermitiana $A$ de dimensão $n$ com autovalores $\lambda_1, \ldots, \lambda_m$ e respectivos autoespaços $H_{\lambda_1}, \ldots, H_{\lambda_m}$. Existe uma base ortonormal $B_{\lambda_i}$ para cada autoespaço $H_{\lambda_i}$, $1 \leq i \leq m$. Então $B = \cup_{i=1}^m B_{\lambda_i}$ é uma base ortonormal de $H$, pelos resultados apresentados acima.

**Teorema A.17 (Representação Espectral).** *Seja $A$ uma matriz hermitiana e $\lambda_1, \ldots, \lambda_k$ seus autovalores distintos. Então*

$$A = \sum_{i=1}^{k} \lambda_i P_i,$$

*onde $P_i$ é o projetor sobre o autoespaço correspondente a $\lambda_i$.*

*Demonstração.* Seja $\phi$ um vetor e considere $\phi_j = P_j\phi$, ou seja, $\phi_j$ é a projeção de $\phi$ sobre o autoespaço correspondente ao autovalor $\lambda_j$. Segue que $\phi = \sum_{j=1}^k \phi_j$, pois os subespaços são ortogonais. Assim,

$$A\phi = A\left(\sum_{j=1}^{k}\phi_j\right) = \sum_{j=1}^{k} A\phi_j = \sum_{j=1}^{k}\lambda_j\phi_j = \sum_{j=1}^{k}\lambda_j P_j\phi$$

e isso completa a prova. □

## A.6 Produto tensorial

**Definição A.18.** *Sejam $H_1, H_2$ espaços de Hilbert com respectivas bases ortonormais $B_1 = \{|\phi_1\rangle, \ldots, |\phi_n\rangle\}$ e $B_2 = \{|\psi_1\rangle, \ldots, |\psi_m\rangle\}$. O* produto tensorial *de $H_1$ e $H_2$, denotado por $H_1 \otimes H_2$, é o espaço de Hilbert $H$ gerado pelo conjunto*

$$B := B_1 \times B_2 = \Big\{\big(|\phi_i\rangle, |\psi_j\rangle\big) : |\phi_i\rangle \in B_1, |\psi_j\rangle \in B_2\Big\},$$

*e com produto interno entre* $\big(|\phi_i\rangle, |\psi_j\rangle\big)$ *e* $\big(|\phi_k\rangle, |\psi_l\rangle\big)$ *dado por* $\langle\phi_i|\phi_k\rangle\langle\psi_j|\psi_l\rangle$ *para quaisquer* $\phi_i, \phi_k \in B_1$ *e* $\psi_j, \psi_l \in B_2$.

Da definição, os elementos de $B$ são ortonormais, de forma que $B$, o conjunto gerador, é também uma base para $H_1 \otimes H_2$ e, portanto, $\dim(H_1 \otimes H_2) = \dim H_1 \dim H_2$. O produto interno entre quaisquer elementos de $H_1 \otimes H_2$ é totalmente determinado pelo produto interno entre os elementos dessa base, que foi definido acima.

**Definição A.19.** *Sejam* $H_1$, $H_2$ *e* $B_1$, $B_2$ *como na definição acima,* $\phi = \sum_{i=1}^{n} c_i|\phi_i\rangle \in H_1$ *e* $\psi = \sum_{j=1}^{m} d_j|\psi_j\rangle \in H_2$. *Definimos o* produto tensorial *de* $|\phi\rangle$ *e* $|\psi\rangle$, *pertencente a* $H_1 \otimes H_2$, *por*

$$|\phi\rangle \otimes |\psi\rangle := \sum_{i=1}^{n}\sum_{j=1}^{m} c_i d_j \big(|\phi_i\rangle, |\psi_j\rangle\big).$$

Algumas vezes será mais conveniente escrevermos $|\phi\rangle|\psi\rangle$ ou $|\phi\psi\rangle$ ou $|\phi, \psi\rangle$ no lugar de $|\phi\rangle \otimes |\psi\rangle$, para $|\phi\rangle \in H_1$ e $|\psi\rangle \in H_2$,

Como observado acima, o produto interno entre os elementos de $B_1$ e $B_2$ determina o produto interno entre quaisquer elementos de $H_1 \otimes H_2$. Em particular, pode-se provar facilmente que o produto interno entre $|\phi\rangle \otimes |\psi\rangle$ e $|\phi'\rangle \otimes |\psi'\rangle$, isto é, $\langle\phi, \psi|\phi', \psi'\rangle$, é dado por $\langle\phi|\phi'\rangle\langle\psi|\psi'\rangle$, para quaisquer $|\phi\rangle, |\phi'\rangle \in H_1$ e $|\psi\rangle, |\psi'\rangle \in H_2$.

Seguem as seguintes propriedades do produto tensorial, para quaisquer vetores $|\phi\rangle \in H_1, |\psi\rangle \in H_2, |\gamma\rangle \in H_3$ e $c \in \mathbb{C}$:

1. $\big(|\phi\rangle \otimes |\psi\rangle\big) \otimes |\gamma\rangle = |\phi\rangle \otimes \big(|\psi\rangle \otimes |\gamma\rangle\big)$ (propriedade associativa)
2. $c\big(|\phi\rangle \otimes |\psi\rangle\big) = \big(c|\phi\rangle\big) \otimes |\psi\rangle = |\phi\rangle \otimes \big(c|\psi\rangle\big)$
3. $\big(|\phi\rangle + |\psi\rangle\big) \otimes |\gamma\rangle = |\phi\rangle \otimes |\gamma\rangle + |\psi\rangle \otimes |\gamma\rangle$ (propriedade distributiva à esquerda)
4. $|\phi\rangle \otimes \big(|\psi\rangle + |\gamma\rangle\big) = |\phi\rangle \otimes |\psi\rangle + |\phi\rangle \otimes |\gamma\rangle$ (propriedade distributiva à direita)

Sejam $B_1, \ldots, B_m$ bases ortonormais dos espaços de Hilbert $H_1, \ldots, H_m$. Então o conjunto

$$B_1 \otimes \cdots \otimes B_m = \bigotimes_{i=1}^{m} B_i = \{\phi_1 \otimes \cdots \otimes \phi_m : \phi_i \in B_i\}$$

é uma base para o espaço de Hilbert

$$H_1 \otimes \cdots \otimes H_m = \bigotimes_{i=1}^{m} H_i.$$

**Exemplo 1.20** Considere o espaço de Hilbert bidimensional $H_2$ com base $B_2 = \{|0\rangle, |1\rangle\}$, onde

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ e } |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Então o conjunto

$$\bigotimes_{i=1}^{n} B_2 = \left\{ |x_1\rangle \otimes \cdots \otimes |x_n\rangle : x_1 \cdots x_n \in \{0,1\}^n \right\}$$

é uma base ortogonal do espaço de Hilbert

$$H_{2^n} := \bigotimes_{i=1}^{n} H_2,$$

de dimensão $2^n$.

Vamos estender o produto tensorial para as matrizes, correspondentes a operadores lineares. Considere as matrizes

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \text{ e } B = \begin{pmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mm} \end{pmatrix}.$$

O produto tensorial de $A$ e $B$, denotado por $A \otimes B$, é definido por

$$A \otimes B := \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{pmatrix}.$$

Seguem as seguintes propriedades:

1. $(A \otimes B)\big(|\phi\rangle \otimes |\psi\rangle\big) = \big(A|\phi\rangle\big) \otimes \big(B|\psi\rangle\big)$
2. $(A \otimes B)(C \otimes D) = AC \otimes BD$
3. $(A \otimes B)^* = A^* \otimes B^*$.

# Apêndice B

# Mecânica quântica

A seguir relacionamos os conceitos de espaços de Hilbert, vistos no capítulo anterior, com seus equivalentes na mecânica quântica.

## B.1 Axiomas

Um *sistema quântico* é representado por um espaço de Hilbert $H$ com a dimensão escolhida apropriadamente. Para nós, esta dimensão sempre será finita. Se tivermos dois sistemas quânticos com seus correspondentes espaços de Hilbert $H_1$ e $H_2$, então o espaço de Hilbert correspondente à composição destes sistemas será $H := H_1 \otimes H_2$.

### B.1.1 Estados quânticos

Antes de definirmos estados quânticos, precisamos introduzir o conceito de raios. Considere a relação $\sim_{\mathrm{R}}$ em $H$ definida da seguinte maneira: dados $|\phi\rangle, |\psi\rangle \in H$, dizemos que $|\phi\rangle \sim_{\mathrm{R}} |\psi\rangle$ se, e somente se, $|\phi\rangle = \alpha|\psi\rangle$ para algum $\alpha \in \mathbb{C}$. É muito fácil ver que a relação $\sim_{\mathrm{R}}$ é reflexiva, simétrica e transitiva, e é portanto uma relação de equivalência em $H$. Então a relação $\sim_{\mathrm{R}}$ induz naturalmente uma partição de $H$ em classes de equivalência. Um *raio (ray)* de $H$ é uma classe de equivalência de $H$ induzida pela relação $\sim_{\mathrm{R}}$. Em outras palavras, um raio é uma classe de equivalência de vetores de $H$ que diferem por um fator multiplicativo complexo.

Dado um sistema quântico representado pelo espaço de Hilbert $H$, um *estado quântico* deste sistema é um raio de $H$. Como um raio é uma classe de equivalência induzida pela relação $\sim_{\mathrm{R}}$, podemos convencionar que utilizaremos como *representantes* de cada classe um vetor de norma unitária. De acordo com esta convenção, se $|\phi\rangle$ é um representante de uma classe, então $|\psi\rangle = e^{i\theta}|\phi\rangle$, para qualquer $\theta \in \mathbb{R}$, também é um representante da mesma classe, ou seja, ambos constituem o mesmo estado quântico. Neste caso, identificamos $|\phi\rangle$ com $|\psi\rangle$, já que eles são fisicamente indistingüíveis, e chamamos o fator $e^{i\theta}$ de *fator de fase global*.

Como cada raio corresponde a um estado quântico possível, dados dois estados quânticos $|\phi\rangle$ e $|\psi\rangle$, ortogonais entre si, podemos formar um novo estado $|\gamma\rangle = \alpha|\phi\rangle + \beta|\psi\rangle$, onde $\alpha, \beta \in \mathbb{C}$ e $\|\alpha\|^2 + \|\beta\|^2 = 1$. O estado $|\gamma\rangle$ é uma superposição dos estados $|\phi\rangle$ e $|\psi\rangle$. Retomando a discussão anterior, os vetores $|\gamma\rangle$ e $e^{i\theta}|\gamma\rangle$ representam o mesmo estado quântico. Já se formarmos o estado $|\varphi\rangle = \alpha|\phi\rangle + e^{i\theta}\beta|\psi\rangle$, com $\theta \in (0, 2\pi)$, não podemos identificar $|\gamma\rangle$ com $|\varphi\rangle$. O fator $e^{i\theta}$ é chamado de *fator de fase relativa* entre $|\gamma\rangle$ e $|\varphi\rangle$ e, diferente do fator de fase global, é importante fisicamente.

Fixada uma base ortonormal $B$ de $H$, chamamos de *estados básicos* de $H$ os estados desta base. Todo estado $|\psi\rangle$ em $H$ pode ser escrito como uma superposição de estados básicos, ou seja,

$$|\psi\rangle = \sum_{|\phi\rangle \in B} \alpha_\phi |\phi\rangle,$$

onde $\alpha_\phi \in \mathbb{C}$ para todo $|\phi\rangle \in B$. O fator $\alpha_\phi$ é chamado de *amplitude* do estado básico $|\phi\rangle$ em $|\psi\rangle$.

A *evolução* de um estado quântico em $H$ durante um certo intervalo de tempo é determinada por uma matriz unitária em $H$. Uma matriz $U$ é unitária se $U^* = U^{-1}$. Uma matriz que define uma possível evolução em um sistema quântico será chamada de *operador*.

### B.1.2 Observáveis e medições

Informalmente, um *observável* é uma propriedade de um sistema quântico que pode ser medida. Formalmente, definimos um *observável* como um operator auto-adjunto em $H$.

De acordo com o teorema A.17 da representação espectral, todo observável $O$ pode ser escrito como

$$O = \sum_{\lambda \in \Lambda} \lambda P_\lambda, \tag{B.1}$$

onde $\Lambda$ é o espectro de $O$ e, para $\lambda \in \Lambda$, $P_\lambda$ é o projetor sobre $W_\lambda$, o autoespaço associado a $\lambda$. A proposição A.16 nos garante que os autoespaços de $O$ são ortogonais, de modo que $H = \bigoplus_{\lambda\in\Lambda} W_\lambda$. Algumas vezes será mais conveniente representarmos um observável pelo seu conjunto de autoespaços. Nesta nova representação, o observável $O$ seria dado por $\{W_\lambda : \lambda \in \Lambda\}$.

Na mecânica quântica, medições são fundamentalmente diferentes de medições na física clássica: uma medição altera irreversivelmente o estado quântico observado. Além disso, o resultado das medições é probabilístico.

Medições são feitas relativas a algum observável. O resultado numérico de uma medição com relação a um observável $O$ é um autovalor de $O$. Considere a representação espectral de $O$, dada em (B.1). Para qualquer autovalor $\lambda$ de $O$, a probabilidade de obtenção de $\lambda$ como resultado da medição de um estado quântico $|\psi\rangle$ é dada por $\left\|P_\lambda|\psi\rangle\right\|^2 = \langle\psi|P_\lambda|\psi\rangle$. Se $\lambda$ é o resultado obtido, então o estado quântico (normalizado) do sistema imediatamente após a medição será

$$\frac{P_\lambda|\psi\rangle}{\sqrt{\langle\psi|P_\lambda|\psi\rangle}},$$

onde o denominador é simplesmente um fator de normalização. Se uma medição relativa ao mesmo observável for repetida imediatamente após a primeira medição, então o resultado será o mesmo, com probabilidade 1.

## B.2 Experimento com polarização de fótons

Vamos exemplificar um experimento simples para nos familiarizarmos com alguns fenômenos quânticos.

Os equipamentos utilizados são uma fonte de luz, como laser, e três polaróides (filtros de polarização) $A$, $B$ e $C$, onde $A$ é polarizado horizontalmente, $B$ a 45° e $C$ verticalmente. Eles devem ser dispostos como indicado na Figura B.1 (inicialmente não utilizaremos $B$):

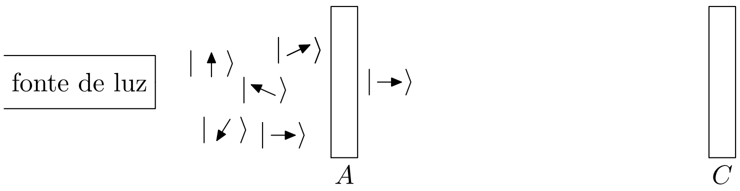


Figura B.1: Disposição do equipamento.

Um fóton pode estar polarizado verticalmente (representado por $|\uparrow\rangle$), horizontalmente (representado por $|\rightarrow\rangle$), ou numa superposição destes estados, ou seja, $\alpha|\uparrow\rangle + \beta|\rightarrow\rangle$, com $\alpha, \beta \in \mathbb{C}$. Assumindo que a luz gerada é aleatoriamente polarizada, cerca de metade dos fótons emitidos serão polarizados horizontalmente por $A$ e o atravessam. Note que $A$ não deixa passar apenas os fótons com $\alpha = 0$ e $\beta = 1$, senão apenas uma fração mínima dos fótons atravessaria $A$, o que não é o caso.

Os fótons polarizados horizontalmente não passam pelo filtro vertical $C$. Porém, se adicionarmos $B$ entre $A$ e $C$, observaremos que 1/8 dos fótons emitidos pela fonte atravessam $C$ (veja a Figura B.2).

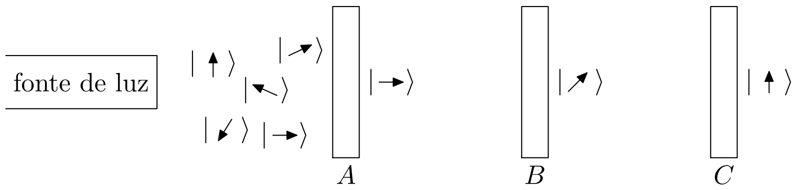


Figura B.2: Nova disposição, utilizando o polaróide $B$.

A mecânica quântica explica esse fenômeno do seguinte modo. Em $A$, ocorre uma medição do fóton, de acordo com o observável $O_A = \{E_1, E_2\}$, onde $E_1$ e $E_2$ são gerados, respectivamente, por $|\uparrow\rangle$ e por $|\rightarrow\rangle$. Após a medição de um fóton no estado de polarização $\alpha|\uparrow\rangle + \beta|\rightarrow\rangle$, ele estará no estado $|\uparrow\rangle$ com probabilidade $|\alpha|^2$ e no estado $|\rightarrow\rangle$ com probabilidade $|\beta|^2$. Apenas estes últimos, projetados sobre $E_2$, atravessam $A$.

Em $C$ ocorre uma medição de acordo com o mesmo observável $O$, porém apenas os fótons projetados sobre $E_1$ pela medição atravessam o polaróide. Se $B$ não estiver entre $A$ e $C$, todos os fótons chegando a $C$ são da forma $0|\uparrow\rangle + 1|\rightarrow\rangle$ e portanto não é possível que nenhum deles o atravesse.

Entretanto, caso $B$ seja interposto entre $A$ e $C$, ele realizará uma medição de acordo com o observável $O = \{E_1', E_2'\}$, onde $E_1'$ e $E_2'$ são gerados respectivamente por $|\nearrow\rangle$ e por $|\searrow\rangle$, onde

$$|\nearrow\rangle := \frac{1}{\sqrt{2}}\big(|\uparrow\rangle + |\rightarrow\rangle\big) \quad \text{e} \quad |\searrow\rangle := \frac{1}{\sqrt{2}}\big(|\uparrow\rangle - |\rightarrow\rangle\big).$$

Com isso, metade dos fótons polarizados horizontalmente que atingem $B$ é projetada sobre $E_1'$ e o atravessa, enquanto a outra metade é projetada sobre $E_2'$ e é absorvida ou refletida.

Agora os fótons que chegam a $C$ não terão mais $\alpha = 0$ e $\beta = 1$, e sim $\alpha = \beta = 1/\sqrt{2}$. Desta forma, metade deles será projetada sobre $E_1$ e atravessará $C$ e a outra metade é absorvida ou refletida. Assim, cerca de 1/8 do total de fótons emitidos atravessa $A$, $B$ e $C$, como observado experimentalmente.

# Apêndice C

# Teoria dos números

Fazemos uma breve revisão de alguns conceitos e resultados básicos de teoria dos números. A familiarização com os tópicos aqui cobertos é fundamental para a compreensão do algoritmo de Shor.

Alguns resultados serão enunciados sem prova. O leitor interessado pode consultar o capítulo 31 do livro de Cormen et al. [CLRS01], o capítulo 4 do livro de Graham et al. [GKP94], os capítulos 10 e 11 do livro de Papadimitriou [Pap94], e os livros de Fraleigh [Fra89], Rosen [Ros00], Coutinho [Cou00] e Bressoud [Bre89].

## C.1 Divisibilidade e primalidade

A divisibilidade de um inteiro por outro é um conceito que permeia toda a teoria dos números.

**Definição C.1.** *Sejam $n$ e $d$ inteiros. Dizemos que $d$* divide *$n$ se existe um inteiro $k$ tal que $n = kd$, e denotamos isso por $d \mid n$. Neste caso, dizemos também que $n$ é* divisível *por $d$ e que $n$ é um* múltiplo *de $d$. Se $d \mid n$ e $d \geq 0$, então dizemos que $d$ é um* divisor *de $n$. Se $d$ não divide $n$, isso é denotado por $d \nmid n$.*

É claro que todo inteiro $n$ é divisível por 1 e $n$, que são chamados de *divisores triviais* de $n$. Um divisor não-trivial de $n$ é chamado de *fator* de $n$.

O seguinte resultado é imediato da definição de divisibilidade:

**Proposição C.2.** *Sejam $d, a, b$ inteiros. Se $d$ divide $a$ e $b$, então $d$ divide $ax+by$ para quaisquer $x, y \in \mathbb{Z}$.*

Também é claro que, se $a$ divide $b$, então $|a| \leq |b|$. Assim, $a \mid b$ e $b \mid a$ implicam que $a = \pm b$.

Os inteiros não-negativos que não têm nenhum fator são extremamente importantes:

**Definição C.3.** *Seja $n > 1$ um inteiro. Dizemos que $n$ é* primo *se os únicos divisores de $n$ são* 1 *e $n$. Caso contrário, dizemos que $n$ é* composto.

Observe que 1 não é nem primo nem composto.

O seguinte fato é conhecido desde os tempos de Euclides:

**Teorema C.4.** *Há infinitos números primos.*

*Demonstração.* Suponha que o conjunto $P$ dos números primos é finito. Considere o inteiro $M := 1 + \prod_{p \in P} p$, ou seja, $M$ é o produto de todos os primos, somado com 1. É óbvio que $M \notin P$, de modo que $M$ deve ser composto, ou seja, $M$ tem um fator primo.

Seja $p$ um primo. É claro que $p$ não divide $M$, pois $p$ divide $M - 1$. Então $M$ não tem nenhum fator primo, um absurdo.

Segue que existem infinitos números primos. □

A operação de divisão e módulo pode ser formalizada para os inteiros através do seguinte resultado:

**Teorema C.5 (Divisão).** *Sejam $m \geq 0$ e $n > 0$ inteiros. Então existem inteiros $q$ e $r$ satisfazendo $m = qn + r$ e $0 \leq r < n$, e tais inteiros são únicos.*

No teorema acima, o valor $q$ é chamado de *quociente* da divisão de $m$ por $n$ e é denotado por $\lfloor m/n \rfloor$ e o número $r$ é o *resto* da divisão de $m$ por $n$, denotado por $m \bmod n$. É óbvio que $n$ divide $m$ se, e somente se, $m \bmod n = 0$.

**Definição C.6.** *Dados inteiros $a, b, d$, dizemos que $d$ é* divisor comum *de $a$ e $b$ se $d$ é divisor de $a$ e de $b$. Se $a \neq 0$ ou $b \neq 0$, então o maior dos divisores comuns de $a$ e $b$ é chamado de* máximo divisor comum *de $a$ e $b$ e é denotado por* $\mathrm{mdc}(a, b)$. *Se* $\mathrm{mdc}(a, b) = 1$, *então dizemos que $a$ e $b$ são* relativamente primos *ou* primos entre si.

Provamos a seguir algumas propriedades importantes sobre divisores comuns.

**Teorema C.7.** *Sejam $a, b$ inteiros, não ambos nulos. Então* $\mathrm{mdc}(a, b)$ *é o menor elemento positivo do conjunto $\{ax + by : x, y \in \mathbb{Z}\}$ de combinações lineares inteiras de $a$ e $b$.*

*Demonstração.* Seja $d$ o menor elemento positivo de $I := \{ax + by : x, y \in \mathbb{Z}\}$ e sejam $x, y \in \mathbb{Z}$ tais que $d = ax + by$.

Vamos mostrar que $\mathrm{mdc}(a, b) \geq d$. Pelo teorema C.5 da divisão, existem inteiros $q$ e $r$ satisfazendo $a = qd + r$ e $0 \leq r < d$. Mas então

$$r \;=\; a - qd = a - q(ax + by) = a(1 - qx) + b(-qy)$$

também é uma combinação linear inteira de $a$ e $b$. Como $d$ é o menor elemento positivo de $I$ e $0 \leq r < d$, então $r = 0$. Segue que $d \mid a$. Analogamente provamos que $d \mid b$. Portanto $\mathrm{mdc}(a, b) \geq d$, pois $d$ é divisor comum de $a$ e $b$.

Agora mostramos que $\mathrm{mdc}(a, b) \leq d$, completando a prova. Pela proposição C.2, $\mathrm{mdc}(a, b)$ divide $d = ax + by$. Como $d > 0$, então $\mathrm{mdc}(a, b) \leq d$, como queríamos. □

**Corolário C.8.** *Sejam $a$ e $b$ inteiros, não ambos nulos. Se $d$ é um divisor comum de $a$ e $b$, então $d \mid \mathrm{mdc}(a, b)$.*

*Demonstração.* Pelo teorema C.7, existem inteiros $x, y$ tais que $\mathrm{mdc}(a, b) = ax + by$, ou seja, $\mathrm{mdc}(a, b)$ é uma combinação linear inteira de $a$ e $b$. Mas $d \mid a$ e $d \mid b$, de modo que $d \mid \mathrm{mdc}(a, b)$, pela proposição C.2. □

O seguinte resultado fornece imediatamente um algoritmo para o cálculo do máximo divisor comum, como veremos posteriormente.

**Teorema C.9 (Recursão de Euclides).** *Sejam $a \geq 0$ e $b > 0$ inteiros. Então*

$$\mathrm{mdc}(a, b) = \mathrm{mdc}(b, a \bmod b). \tag{C.1}$$

*Demonstração.* Seja $d := \mathrm{mdc}(a, b)$ e sejam $q := \lfloor a/b \rfloor$ e $r := a \bmod b$. Vamos mostrar que $\mathrm{mdc}(a, b) \mid \mathrm{mdc}(b, r)$ e que $\mathrm{mdc}(b, r) \mid \mathrm{mdc}(a, b)$, de onde seguirá o teorema.

É claro que $d \mid a$ e $d \mid b$. Como $r = a - bq$ é uma combinação linear inteira de $a$ e $b$, segue da proposição C.2 que $d \mid r$. Então $d \mid b$ e $d \mid r$, ou seja, $d$ é um divisor comum de $b$ e $r$. Pelo corolário C.8, $d \mid \mathrm{mdc}(b, r)$, ou seja, $\mathrm{mdc}(a, b) \mid \mathrm{mdc}(b, r)$.

O outro lado é análogo, pois $a = qb + r$ é uma combinação linear inteira de $b$ e $r$. □

**Proposição C.10.** *Sejam $d, a$ e $b$ inteiros positivos e suponha que $d \mid ab$. Se $\mathrm{mdc}(d, a) = 1$, então $d \mid b$.*

*Demonstração.* Como $\mathrm{mdc}(d, a) = 1$, o teorema C.7 nos garante que existem inteiros $x, y$ tais que $dx + ay = 1$. Multiplicando esta equação por $b$, temos $bdx + aby = b$. É óbvio que $d \mid bdx$, a primeira parcela. Por hipótese, $d \mid aby$, a segunda parcela. Mas então $d \mid b$ pela proposição C.2, como queríamos. □

O seguinte resultado segue imediatamente da proposição C.10 acima.

**Teorema C.11.** *Sejam $p$ um primo e $a, b$ inteiros. Se $p \mid ab$, então $p \mid a$ ou $p \mid b$.*

O teorema C.11 pode ser utilizado para provar o seguinte resultado, conhecido como Teorema Fundamental da Aritmética:

**Teorema C.12 (Fatoração única).** *Seja $n > 1$ um inteiro. Então existe um único modo de escrever $n$ na forma*

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}, \tag{C.2}$$

*onde $k \geq 1$, os inteiros $p_1 < p_2 < \cdots < p_k$ são primos e $e_i > 0$ para todo $i$.*

Na equação (C.2), o lado direito é chamado de *fatoração de $n$ em primos.*

## C.2 Teoria dos grupos

**Definição C.13.** *Sejam $G \subseteq G'$ conjuntos e $* : G' \times G' \longrightarrow G'$ uma função. Um* grupo *$(G, *)$ é um conjunto $G$ munido de uma função $*$ satisfazendo os seguintes axiomas:*

1. Fechamento: *$a * b \in G$ para todo $a, b \in G$.*
2. Associatividade: *$(a * b) * c = a * (b * c)$ para todos $a, b, c \in G$.*
3. Existência de identidade: *Existe $e \in G$ tal que $a * e = e * a = a$ para todo $a \in G$. Tal elemento $e$ é chamado de* elemento identidade *do grupo $(G, *)$.*
4. Existência de inversos: *Para todo $a \in G$, existe $b \in G$ tal que $a * b = b * a = e$. Tal elemento $b$ é chamado de* inverso *de $a$ e é denotado por $a^{-1}$.*

*Se $(G, *)$ é um grupo, então a função $*$ é chamada de* operação binária *do grupo.*

*Se, além desses axiomas, o par $(G, *)$ satisfizer $a * b = b * a$ para todo $a, b \in G$, então dizemos que $(G, *)$ é um grupo* comutativo *ou* abeliano.

*Dizemos que um grupo $(G, *)$ é finito se $G$ é finito. Neste caso, a* ordem de *$(G, *)$ é $\operatorname{ord}\big((G, *)\big) := |G|$.*

Acima, definimos um grupo como um par ordenado $(G, *)$ formado por um conjunto $G$ e uma operação binária $*$. Em alguns momentos abusaremos da notação e diremos que $G$ é um grupo, quando a operação binária $*$ estiver implícita no contexto.

É óbvio que $(\mathbb{Z}, +)$ é um grupo comutativo, tendo 0 como identidade e $-n$ como inverso de $n$ para todo $n \in \mathbb{Z}$. É claro também que $(\mathbb{Z}, \cdot)$ não é um grupo, pois, por exemplo, o elemento 2 não tem inverso multiplicativo nos inteiros.

Vamos construir, para cada inteiro positivo $n$, dois grupos finitos, através das operações de adição e multiplicação módulo $n$.

Fixe $n$ um inteiro positivo. Considere a relação $\sim_n \subseteq \mathbb{Z} \times \mathbb{Z}$ definida da seguinte maneira: dados $a, b \in \mathbb{Z}$, dizemos que $a \sim_n b$ se, e somente se, $n$ divide $a - b$. Se $a \sim_n b$, então dizemos que $a$ e $b$ são *congruentes módulo $n$*.

É muito fácil provar que a relação $\sim_n$ é reflexiva, simétrica e transitiva, e é portanto uma relação de equivalência em $\mathbb{Z}$. Então a relação $\sim_n$ induz naturalmente uma partição de $\mathbb{Z}$ em classes de equivalência. Dados inteiros $a, b$, é fácil provar que $a$ e $b$ estão na mesma classe de equivalência se, e somente se, $a \bmod n = b \bmod n$, isto é, se $a$ e $b$ têm o mesmo resto na divisão por $n$. Portanto, a classe de equivalência contendo o inteiro $a$ é $[a]_n := \{a + kn : k \in \mathbb{Z}\}$.

Definimos $\mathbb{Z}_n$ como o conjunto de classes de equivalência induzidas pela relação $\sim_n$, ou seja, $\mathbb{Z}_n := \big\{[a]_n : 0 \leq a < n\big\}$. Podemos convencionar que o representante de uma classe $[a]_n$ é o único elemento não-negativo da classe que é estritamente menor que $n$. Mas é importante lembrar que cada elemento de $\mathbb{Z}_n$ é uma classe de equivalência: quando nos referirmos, por exemplo, ao elemento $-1$ de $\mathbb{Z}_n$, estamos nos referindo à classe representada por $n - 1$, de acordo com a nossa convenção.

Podemos definir agora as operações de adição e multiplicação módulo $n$. Seja $+_n : \mathbb{Z}_n \times \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ definida da seguinte forma: dados $[a]_n, [b]_n \in \mathbb{Z}_n$, tomamos $[a]_n +_n [b]_n := [a + b]_n$. A multiplicação módulo $n$ é definida de forma análoga: dados $[a]_n, [b]_n \in \mathbb{Z}_n$, tomamos $[a]_n \cdot_n [b]_n := [ab]_n$. Com as operações definidas deste modo, dados representantes $a$ e $b$, para obtermos o representante da adição e multiplicação módulo $n$ de $a$ e $b$, basta aplicarmos a operação desejada sobre $a$ e $b$, vistos como inteiros, e calcular o resto da divisão do resultado por $n$.

É muito fácil provar o seguinte resultado:

**Proposição C.14.** *O par $(\mathbb{Z}_n, +_n)$ é um grupo comutativo finito.*

Chamamos o grupo $(\mathbb{Z}_n, +_n)$ de *grupo aditivo módulo $n$*. Escreveremos, de agora em diante, $+$ no lugar de $+_n$.

Vamos definir agora o *grupo multiplicativo módulo $n$*. Considere o conjunto

$$\mathbb{Z}_n^* := \big\{[a]_n : \mathrm{mdc}(a, n) = 1\big\}. \tag{C.3}$$

**Teorema C.15.** *O par $(\mathbb{Z}_n^*, \cdot_n)$ é um grupo comutativo finito.*

*Demonstração.* É fácil provar que valem as propriedades associativa, comutativa e de fechamento e que $\mathbb{Z}_n^*$ é finito. A identidade é $[1]_n$. Resta provarmos a existência de inversos. Seja $[a]_n \in \mathbb{Z}_n^*$. Como $\mathrm{mdc}(a, n) = 1$, o teorema C.7 nos garante que existem inteiros $x, y$ tais que $ax + ny = 1$. Mas então $ax \sim_n 1$, isto é, $[a]_n \cdot_n [x]_n = [1]_n$ e portanto $[x]_n$ é o inverso de $[a]_n$. □

De agora em diante passamos a escrever $\cdot$ no lugar de $\cdot_n$ ou simplesmente omitimos o símbolo da operação, que ficará clara pela justaposição dos fatores. Escreveremos também $a \equiv b \pmod{n}$ no lugar de $a \sim_n b$ e de $[a]_n = [b]_n$.

Vamos abusar da notação e chamar $\mathbb{Z}_n$ de grupo aditivo módulo $n$ e $\mathbb{Z}_n^*$ de grupo multiplicativo módulo $n$.

A função $\phi(n) := |\mathbb{Z}_n^*|$ é chamada de *função totiente de Euler* e conta o número de inteiros entre 0 e $n - 1$, inclusive, que são relativamente primos a $n$. Mais adiante mostraremos uma forma de calcular $\phi(n)$ a partir dos divisores primos de $n$.

Definimos a seguir o conceito de subgrupo:

**Definição C.16.** *Sejam $(G, *)$ um grupo e $H \subseteq G$. Se $(H, *)$ é um grupo, então $(H, *)$ é chamado de* subgrupo de $(G, *)$.

É claro que o grupo $(2\mathbb{Z}, +)$, onde $2\mathbb{Z} := \{2n : n \in \mathbb{Z}\}$ é o conjunto dos pares, é um subgrupo de $(\mathbb{Z}, +)$. Na verdade, para qualquer $k \in \mathbb{Z}$, o grupo $(k\mathbb{Z}, +)$, onde $k\mathbb{Z} := \{kn : n \in \mathbb{Z}\}$, é subgrupo de $(\mathbb{Z}, +)$.

O seguinte teorema é um poderoso resultado da álgebra. Sua demonstração é muito simples e pode ser vista no livro de Fraleigh [Fra89].

**Teorema C.17 (Lagrange).** *Sejam $(G, *)$ um grupo finito e $(H, *)$ um subgrupo de $(G, *)$. Então $|H|$ é um divisor de $|G|$.*

Se $(H, *)$ é subgrupo de $(G, *)$ e $H \neq G$, então $(H, *)$ é chamado de *subgrupo próprio de $G$*. O corolário a seguir é imediato do teorema C.17 de Lagrange.

**Corolário C.18.** *Se $(H, *)$ é um subgrupo próprio de um grupo finito $(G, *)$, então temos $|H| \leq |G|/2$.*

Seja $(G, *)$ um grupo finito e seja $g \in G$. Vamos mostrar uma maneira sistemática de se obter o "menor" subgrupo de $(G, *)$ contendo $g$. Ou seja, vamos construir um subgrupo $(H, *)$ de $(G, *)$ tal que $g \in H$ e, para todo subgrupo $(H', *)$ de $(G, *)$ contendo $g$, temos $H \subseteq H'$.

Para tanto, começamos encontrando todos os elementos de $G$ que podem ser obtidos a partir da aplicação da operação $*$ em $g$. Seja $e \in G$ o elemento identidade do grupo $(G, *)$. Para $k \geq 0$, defina $g^k$ como

$$g^k := \begin{cases} e & \text{se } k = 0, \\ g * g^{k-1} & \text{se } k > 0. \end{cases} \tag{C.4}$$

É óbvio que todo grupo contendo $g$ também deve conter $g^k$, para todo $k \geq 0$. Mas o inverso de $g$ também deve estar nesse grupo. Para $k \leq -1$, defina $g^k$ como

$$g^k := \begin{cases} g^{-1} & \text{se } k = -1, \\ g^{-1} * g^{k+1} & \text{se } k < -1. \end{cases} \tag{C.5}$$

É muito fácil provar o seguinte teorema:

**Teorema C.19.** *Seja $(G, *)$ um grupo e seja $g \in G$. Então $(H, *)$, onde*

$$H := \{g^k : k \in \mathbb{Z}\}, \tag{C.6}$$

*é um subgrupo de $(G, *)$.*

Denotamos o conjunto $H$ do teorema C.19 por $\langle g \rangle$, isto é, $\langle g \rangle := \{g^k : k \in \mathbb{Z}\}$. É claro que $\langle g \rangle$ é o menor subgrupo de $(G, *)$ contendo $g$, no sentido que foi discutido anteriormente.

**Definição C.20.** *Sejam $G$ um grupo e $g \in G$. O subgrupo $\langle g \rangle$ é o* subgrupo gerado por $g$. *Dizemos também que $g$* gera *o subgrupo $\langle g \rangle$ ou que $g$ é o* gerador *do subgrupo $\langle g \rangle$. Se $H$ é um subgrupo de $G$ e $H = \langle g \rangle$, então $H$ é o* subgrupo cíclico gerado por $g$. *Se $G = \langle g \rangle$, então $G$ é um* grupo cíclico.

Se $H$ é um subgrupo cíclico gerado por $g$, a ordem de $H$, usualmente denotada por $\operatorname{ord}(H)$, poderá ser alternativamente denotada por $\operatorname{ord}(g)$. O valor $\operatorname{ord}(g)$ é chamado também de *ordem de $g$ em $G$*.

**Teorema C.21.** *Seja $(G, *)$ um grupo. Sejam $e$ o elemento identidade de $(G, *)$ e $g \in G$. Então $\operatorname{ord}(g)$ é o menor inteiro positivo $t$ tal que $g^t = e$.*

*Demonstração.* Seja $t$ o menor inteiro positivo tal que $g^t = e$. Seja $k \in \mathbb{Z}$. Pelo teorema C.5 da divisão, existem inteiros $q, r$ tais que $k = qt + r$ e $0 \leq r < t$. Então $g^k = g^{qt+r} = (g^t)^q * g^r = e^q * g^r = g^r$. Assim, $|\langle g \rangle| \leq t$.

Agora vamos mostrar que $|\langle g \rangle| \geq t$. Suponha por um instante que existem inteiros $1 \leq i < j \leq t$ tais que $g^i = g^j$. Então $g^{i+k} = g^{j+k}$ para todo $k \geq 0$. Mas então $g^t = g^{j+(t-j)} = e$ e, portanto, $g^{i+(t-j)} = e$. Mas isso é um absurdo, pois $0 < i + (t - j) < t$ e $t$ é o menor inteiro positivo satisfazendo $g^t = e$. Segue que $|\langle g \rangle| \geq t$.

Portanto, $\operatorname{ord}(g) = |\langle g \rangle| = t$, como queríamos. $\square$

O seguinte corolário é imediato da prova do teorema C.21:

**Corolário C.22.** *Sejam $G$ um grupo e $g \in G$. Então a seqüência $\langle g^0, g^1, g^2, \ldots \rangle$ é periódica com período $t := \mathrm{ord}(g)$. Em outras palavras, $g^i = g^j$ se, e somente se, $i \equiv j \pmod{t}$.*

Outro poderoso resultado da álgebra pode ser sintetizado a partir do corolário C.22 e do teorema C.17 de Lagrange:

**Corolário C.23.** *Seja $(G, *)$ um grupo e seja $e \in G$ o elemento identidade de $(G, *)$. Então, para todo $g \in G$,*

$$g^{|G|} = e. \tag{C.7}$$

*Demonstração.* Seja $t := \mathrm{ord}(g)$. Pelo teorema C.17 de Lagrange, $t$ divide $|G|$ e, portanto, $|G| \equiv 0 \pmod{t}$. Mas então, pelo corolário C.22, $g^{|G|} = g^0 = e$, como queríamos. □

Vamos agora aplicar estes resultados ao grupo $\mathbb{Z}_n^*$. Reescrevendo o corolário C.23 na notação dos grupos multiplicativos, temos o seguinte teorema:

**Teorema C.24 (Euler).** *Seja $n > 1$. Então, para todo $a \in \mathbb{Z}_n^*$,*

$$a^{\phi(n)} \equiv 1 \pmod{n}. \tag{C.8}$$

Como $\phi(p) = p - 1$ para todo primo $p$, então obtemos imediatamente do teorema C.24 de Euler o resultado a seguir.

**Teorema C.25 (Fermat).** *Seja $p$ um primo. Então, para todo $a \in \mathbb{Z}_p^*$,*

$$a^{p-1} \equiv 1 \pmod{p}. \tag{C.9}$$

**Definição C.26.** *Sejam $n > 1$ e $r \in \mathbb{Z}_n^*$. Se $r$ é gerador do grupo $\mathbb{Z}_n^*$, então $r$ é uma* raiz primitiva de $\mathbb{Z}_n^*$.

Seja $r \in \mathbb{Z}_n^*$. É óbvio que $r$ é uma raiz primitiva de $\mathbb{Z}_n^*$ se, e somente se, $\mathrm{ord}_n(r) = \phi(n)$, onde $\mathrm{ord}_n(r)$ denota a ordem de $r$ no grupo $\mathbb{Z}_n^*$.

Se $r$ é uma raiz primitiva de $\mathbb{Z}_n^*$, então o grupo $\mathbb{Z}_n^*$ é cíclico e todo elemento de $\mathbb{Z}_n^*$ é uma potência de $r$. Ou seja, para todo $a \in \mathbb{Z}_n^*$, existe um $0 \leq z < \phi(n)$ inteiro tal que $r^z \equiv a \pmod{n}$.

**Definição C.27.** *Sejam $n > 1$ e $r$ uma raiz primitiva de $\mathbb{Z}_n^*$. Seja $a \in \mathbb{Z}_n^*$. Sabemos que existe um inteiro $0 \leq z < \phi(n)$ tal que $r^z \equiv a \pmod{n}$. Tal $z$ é o* logaritmo discreto de $a$ módulo $n$ na base $r$, *denotado por* $\mathrm{ind}_{n,r}(a)$.

Não é verdade que $\mathbb{Z}_n^*$ é cíclico para todo $n$:

**Teorema C.28.** *Os valores de $n$ para os quais $\mathbb{Z}_n^*$ é cíclico são 2, 4, $p^k$ e $2p^k$ para todos os primos ímpares $p$ e inteiros positivos $k$.*

## C.3 Teorema do resto chinês

Seja $n$ um número composto e suponha que $n$ pode ser escrito como um produto de inteiros mutuamente primos entre si, ou seja, $n = n_1 \cdots n_k$, com $\operatorname{mdc}(n_i, n_j) = 1$ para todo $i \neq j$. O teorema do resto chinês mostra que o grupo $\mathbb{Z}_n$ tem a "mesma estrutura" do produto cartesiano $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$.

**Teorema C.29 (Resto chinês).** *Seja $n = n_1 \cdots n_k$, onde os $n_i$ são mutuamente primos entre si. Considere a correspondência*

$$a \leftrightarrow (a_1, \ldots, a_k), \tag{C.10}$$

*onde $a \in \mathbb{Z}_n$, $a_i \in \mathbb{Z}_{n_i}$ e $a_i = a \bmod n_i$ para todo $i$. Então a correspondência* (C.10) *é uma bijeção entre $\mathbb{Z}_n$ e o produto cartesiano $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$.*

*Demonstração.* Dado $a \in \mathbb{Z}_n$, é muito fácil encontrar inteiros $a_1, \ldots, a_k$ com $a_i = a \bmod n_i$ e $a_i \in \mathbb{Z}_{n_i}$ para todo $i$: bastam $k$ divisões.

Sejam $a_1, \ldots, a_k$ inteiros com $a_i \in \mathbb{Z}_{n_i}$ para todo $i$. Vamos mostrar como obter $a \in \mathbb{Z}_n$ tal que $a_i = a \bmod n_i$ para todo $i$.

Para cada $i$, defina $m_i := n/n_i$, isto é, $m_i = \prod_{j \neq i} n_j$ é o produto de todos os $n_j$'s exceto $n_i$. É claro que $\operatorname{mdc}(m_i, n_i) = 1$. Pelo teorema C.7, existem inteiros $x, y$ tais que $xm_i + yn_i = 1$, ou seja, $xm_i \equiv 1 \pmod{n_i}$. Defina $m_i^{-1} := x \bmod n_i$, pois $x$ é o inverso multiplicativo de $m_i$ módulo $n_i$. Finalmente, tome $c_i := m_i m_i^{-1} \bmod n_i$.

Seja $j \neq i$. Então $m_j \equiv 0 \pmod{n_i}$, pois $m_i \mid m_j$. Concluímos assim que $c_j \equiv m_j \equiv 0 \pmod{n_i}$. Também temos $c_i \equiv m_i^{-1} m_i \equiv 1 \pmod{n_i}$. Mas então $c_i \leftrightarrow (0, \ldots, 0, 1, 0, \ldots, 0)$, onde todos os componentes são nulos exceto o $i$-ésimo, que vale 1.

Tome $a := a_1 c_1 + \cdots + a_k c_k \bmod n$. Observe agora que, para todo $i$, temos $a \equiv a_i c_i \pmod{n_i}$ e, portanto, $a \equiv a_i \pmod{n_i}$, já que $c_i \equiv 1 \pmod{n_i}$. Segue que a correspondência (C.10) é uma bijeção, como queríamos. □

Os seguintes corolários seguem imediatamente do teorema C.29 do resto chinês.

**Corolário C.30.** *Sejam $n_1, \ldots, n_k$ inteiros mutuamente primos entre si e $n = n_1 \cdots n_k$. Então, para quaisquer inteiros $a_1, \ldots, a_k$, o sistema de equações $x \equiv a_i \pmod{n_i}$ para $i = 1, \ldots, k$ tem uma única solução módulo $n$.*

**Corolário C.31.** *Sejam $n_1, \ldots, n_k$ inteiros mutuamente primos entre si e $n = n_1 \cdots n_k$. Então, para quaisquer inteiros $x$ e $a$, temos $x \equiv a \pmod{n_i}$ para $i = 1, \ldots, k$ se, e somente se, $x \equiv a \pmod{n}$.*

Podemos agora provar uma fórmula para a função totiente de Euler:

**Teorema C.32.** *Seja $n > 1$ um inteiro. Então*

$$\phi(n) = n \prod_{p \in P} \left(1 - \frac{1}{p}\right), \tag{C.11}$$

*onde $P$ é o conjunto dos divisores primos de $n$.*

*Demonstração.* É óbvio que $\phi(p) = p - 1$ para todo primo $p$, o que está de acordo com a equação (C.11).

Suponha agora que $n$ é uma potência de primo, isto é, que $n = p^k$ para algum $k \geq 1$. Então é fácil calcular $\phi(n)$: os únicos inteiros entre 0 e $n-1$, inclusive, que não são relativamente primos a $n$ são os múltiplos de $p$, ou seja, $0, p, 2p, \ldots, (p^{k-1} - 1)p$. Portanto $\phi(n) = n - p^{k-1} = p^k - p^{k-1} = p^k(1 - 1/p)$. Observe novamente que estamos de acordo com a equação (C.11) e que esta fórmula fornece apropriadamente $\phi(p) = p - 1$ quando o expoente $k$ vale 1.

Suponha então que $n$ não é uma potência de primo. Então podemos escrever $n$ como um produto $n = n_1 n_2$ de inteiros $n_1, n_2$ relativamente primos. Então qualquer inteiro $0 \leq m < n$ pode ser representado pelo par $(m \bmod n_1, m \bmod n_2)$ e, pelo corolário C.30, tal representação é única.

Observe agora que $\mathrm{mdc}(m, n) = 1$ se, e somente se, $\mathrm{mdc}(m, n_1) = 1$ e $\mathrm{mdc}(m, n_2) = 1$. Pelo teorema C.9, temos então que $\mathrm{mdc}(m, n) = 1$ se, e somente se, $\mathrm{mdc}(m \bmod n_1, n_1) = 1$ e $\mathrm{mdc}(m \bmod n_2, n_2) = 1$. Portanto podemos calcular $\phi(n)$ recursivamente como $\phi(n) = \phi(n_1)\phi(n_2)$, pois há $\phi(n_1)$ possíveis valores para a primeira coordenada e $\phi(n_2)$ possíveis valores para a segunda coordenada da representação mostrada acima tais que o inteiro representado seja relativamente primo a $n$.

Uma função $f$ sobre inteiros positivos é dita *multiplicativa* se $f(1) = 1$ e $f(ab) = f(a)f(b)$ sempre que $\mathrm{mdc}(a, b) = 1$. Acabamos de provar, então, que a função totiente $\phi(n)$ é multiplicativa. É claro que uma função multiplicativa é completamente definida pelo seu valor nas potências de primos. De fato, suponha que $f$ é multiplicativa e que a fatoração de $m$ seja

$$m = \prod_{p \in P} p^{m_p},$$

onde $P$ é o conjunto dos divisores primos de $m$. Então

$$f(m) = \prod_{p \in P} f(p^{m_p}).$$

Mas nós já sabemos o valor de $\phi(n)$ se $n$ é uma potência de primo. Então temos

$$\phi(n) = \prod_{p \in P} \phi(p^{m_p}) = \prod_{p \in P} \left[ p^{m_p} \left( 1 - \frac{1}{p} \right) \right] = n \prod_{p \in P} \left( 1 - \frac{1}{p} \right),$$

como queríamos. □

## C.4 Equações modulares

Estudaremos agora algumas equações sobre $\mathbb{Z}_n$ e $\mathbb{Z}_n^*$.

Primeiro consideramos as soluções da equação

$$ax \equiv b \pmod{n}, \tag{C.12}$$

onde $a$ e $n$ são inteiros positivos. Vamos denotar por $\langle a \rangle$ o subgrupo de $\mathbb{Z}_n$ gerado por $a$, isto é, $\langle a \rangle := \{ax \bmod n : x > 0\}$. É óbvio que a equação (C.12) admite soluções em $\mathbb{Z}_n$ se, e somente se, $b \in \langle a \rangle$. O teorema a seguir caracteriza o subgrupo $\langle a \rangle$:

**Teorema C.33.** *Sejam $a$ e $n$ inteiros positivos e $d = \mathrm{mdc}(a, n)$. Então*

$$\langle a \rangle = \langle d \rangle = \Big\{ 0, d, 2d, \ldots, \big((n/d) - 1\big) d \Big\} \tag{C.13}$$

*e portanto $|\langle a \rangle| = n/d$.*

*Demonstração.* Primeiro vamos mostrar que $\langle d \rangle \subseteq \langle a \rangle$. Pelo teorema C.7, existem inteiros $x$ e $y$ tais que $ax + ny = d$, ou seja, $ax \equiv d \pmod{n}$. Então $d \in \langle a \rangle$. É claro também que todos os múltiplos de $d$ estão em $\langle a \rangle$. Segue que $\langle d \rangle \subseteq \langle a \rangle$.

Agora mostraremos que $\langle a \rangle \subseteq \langle d \rangle$. Seja $m \in \langle a \rangle$. Então $m \equiv ax \pmod{n}$ para algum $x > 0$. Assim, existe um inteiro $y$ tal que $m = ax + ny$. Como $d \,|\, a$ e $d \,|\, n$, então $d \,|\, m$ pela proposição C.2. Portanto $m \in \langle d \rangle$, como queríamos.

Finalmente, observe que existem exatamente $n/d$ múltiplos de $d$ entre 0 e $n - 1$, de modo que $|\langle a \rangle| = n/d$. □

**Corolário C.34.** *Sejam $a, b$ inteiros, $n$ um inteiro positivo e $d = \mathrm{mdc}(a, n)$. Se $d \,|\, b$, então a equação* (C.12) *admite exatamente $d$ soluções em $\mathbb{Z}_n$. Caso contrário, ela não admite nenhuma solução.*

*Demonstração.* É óbvio que a equação $ax \equiv b \pmod{n}$ admite soluções se, e somente se, $b \in \langle a \rangle$. Pelo teorema C.33, $\langle a \rangle = \langle d \rangle$. Então a equação (C.12) admite solução se, e somente se, $d \,|\, b$.

Resta provarmos que, se $d \mid b$, então a equação (C.12) admite $d$ soluções em $\mathbb{Z}_n$. Suponha, então, que $d \mid b$. Sabemos pelo corolário C.22 que a seqüência $ai \bmod n$, com $i \geq 0$, é periódica com período $\operatorname{ord}(a) = |\langle a \rangle|$ e, pelo teorema C.12, $|\langle a \rangle| = n/d$. Então $b \bmod n$ aparece $d$ vezes na seqüência $ai \bmod n$, com $0 \leq i \leq n-1$: uma vez em cada bloco de comprimento $n/d$. Os índices $x$ para os quais $ax \bmod n = b \bmod n$ são as soluções da equação (C.12) em $\mathbb{Z}_n$. Então há $d$ soluções para a equação (C.12), como queríamos. □

Consideraremos agora alguns resultados interessantes acerca de equações sobre $\mathbb{Z}_n^*$.

**Teorema C.35 (Logaritmo discreto).** *Seja $r$ uma raiz primitiva de $\mathbb{Z}_n^*$. Então a equação*

$$r^x \equiv r^y \pmod{n}$$

*vale se, e somente se, a equação*

$$x \equiv y \pmod{\phi(n)}$$

*vale.*

*Demonstração.* Suponha que $x \equiv y \pmod{\phi(n)}$. Então $x = y + k\phi(n)$ para algum inteiro $k$. Assim,

$$\begin{aligned} r^x &\equiv r^{y+k\phi(n)} && \pmod{n} \\ &\equiv r^y \left(r^{\phi(n)}\right)^k && \pmod{n} \\ &\equiv r^y \cdot 1^k && \pmod{n} \\ &\equiv r^y && \pmod{n}, \end{aligned}$$

onde utilizamos o fato de que $r^{\phi(n)} \equiv 1 \pmod{n}$, pelo teorema C.24 de Euler.

Suponha agora que $r^x \equiv r^y \pmod{n}$. Como $r$ é uma raiz primitiva de $\mathbb{Z}_n^*$, então $|\langle r \rangle| = \phi(n)$. Pelo corolário C.22, a seqüência de potências de $r$ módulo $n$ é periódica com período $\phi(n)$. Mas então, $r^x \equiv r^y \pmod{n}$ se, e somente se, $x \equiv y \pmod{\phi(n)}$. □

**Teorema C.36.** *Seja $p$ um primo ímpar e seja $k \geq 1$ um inteiro. Então a equação*

$$x^2 \equiv 1 \pmod{p^k} \tag{C.14}$$

*tem exatamente duas soluções, a saber, $x = \pm 1$.*

*Demonstração.* Seja $n := p^k$. Pelo teorema C.28, o grupo $\mathbb{Z}_n^*$ tem um raiz primitiva. Seja $r$ uma raiz primitiva de $\mathbb{Z}_n^*$. Utilizando logaritmos discretos, a equação (C.14) pode ser escrita como

$$\left(r^{\operatorname{ind}_{n,r}(x)}\right)^2 \equiv r^{\operatorname{ind}_{n,r}(1)} \pmod{n}. \tag{C.15}$$

Pelo teorema C.35 do logaritmo discreto, a equação (C.15) é equivalente a

$$2 \operatorname{ind}_{n,r}(x) \equiv 0 \pmod{\phi(n)}, \tag{C.16}$$

pois $\operatorname{ind}_{n,r}(1) = 0$.

Pelo teorema C.32, temos $\phi(n) = \phi(p^k) = p^k(1-1/p) = (p-1)p^{k-1}$. Então temos $d = \operatorname{mdc}\big(2, \phi(n)\big) = 2$, pois $p-1$ é par. Como $d \mid 0$, o corolário C.34 nos garante que a equação (C.16) tem exatamente $d = 2$ soluções. Portanto a equação (C.14) admite exatamente 2 soluções. É fácil ver então que $x = \pm 1$ são as únicas soluções da equação (C.14). $\square$

Podemos considerar equações da forma $x^2 \equiv 1 \pmod{n}$ para qualquer $n$ positivo. Soluções diferentes de $\pm 1$ são chamadas de *raízes quadradas não-triviais de* 1 *módulo* $n$.

O corolário a seguir é imediato da contrapositiva do teorema C.36.

**Corolário C.37.** *Seja* $n > 1$ *um inteiro. Se existe uma raiz quadrada não-trivial de* 1*, módulo* $n$*, então* $n$ *é composto.*

Vamos ver agora que, se tivermos à nossa disposição uma raiz quadrada não-trivial de um $n > 0$, então podemos facilmente obter um fator de $n$. Um algoritmo eficiente para o cálculo do máximo divisor comum será apresentado mais adiante.

**Teorema C.38.** *Sejam* $n > 1$ *um inteiro e* $x$ *uma raiz quadrada não-trivial de* 1*, módulo* $n$*. Então ambos* $\operatorname{mdc}(x-1, n)$ *e* $\operatorname{mdc}(x+1, n)$ *são divisores não-triviais de* $n$*.*

*Demonstração.* Sabemos, pelo corolário C.37, que $n$ é composto. Como $x^2 \equiv 1 \pmod{n}$, então $n$ divide $x^2 - 1 = (x+1)(x-1)$. Por outro lado, $x \not\equiv 1 \pmod{n}$ implica que $n \nmid (x-1)$ e $x \not\equiv -1 \pmod{n}$ implica que $n \nmid (x+1)$. Então os fatores de $n$ devem estar separados entre $x-1$ e $x+1$. Logo, ambos $\operatorname{mdc}(x-1, n)$ e $\operatorname{mdc}(x+1, n)$ são fatores de $n$. $\square$

## C.5 Considerações computacionais

Para os algoritmos que resolvem problemas de teoria dos números, adotamos a convenção de que um número $n$ é codificado através de sua representação binária. Assim o tamanho da entrada de um algoritmo que recebe inteiros $n_1, \ldots, n_k$ é $\beta_1 + \cdots + \beta_k$, onde $\beta_i := \lfloor \lg n_i \rfloor + 1 = O(\lg n_i)$ é o comprimento da representação binária de $n_i$ e $\lg m$ denota o logaritmo de $m$ na base 2. Tal algoritmo é polinomial se seu tempo de execução for limitado por um polinômio em $\lg n_1, \ldots, \lg n_k$.

Também precisamos ser cuidadosos com relação ao tempo de execução das operações aritméticas. Para muitos algoritmos, costuma-se considerar somas e multiplicações como *operações elementares* (ou *primitivas*) e que portanto executam em tempo constante. Porém, dado que estaremos trabalhando com números "grandes", o tempo de execução destas operações será relevante.

Como as operações aritméticas para inteiros de tamanho arbitrário são definidas em termos de operações sobre os bits de sua representação binária, é razoável medirmos o tempo de execução dos algoritmos de teoria dos números através do número de *operações sobre bits* que eles realizam. Por exemplo, a soma de dois números de $\beta$ bits pode ser feita da maneira usual, como é feita com lápis e papel, de modo a envolver $O(\beta)$ operações sobre bits. É claro que o tempo de execução deste algoritmo é linear no tamanho da entrada.

A maneira usual de se multiplicar ou dividir, como se faz com lápis e papel, fornece algoritmos que executam em tempo $O(\beta^2)$ quando recebem como entrada inteiros de $\beta$ bits. Mas existem algoritmos mais rápidos para multiplicação. Uma simples aplicação do método de divisão e conquista fornece um algoritmo cujo tempo de execução é $O(\beta^{\lg 3})$, o que foi originalmente sugerido por Karatsuba e Ofman [KO62, KO63]. Atualmente, o algoritmo assintoticamente mais rápido é o de Schönhage e Strassen [SS71, Knu81, Sch82], com tempo de execução $O(\beta \lg \beta \lg \lg \beta)$. Mais detalhes sobre esses algoritmos podem ser vistos no livro de Knuth [Knu81].

O algoritmo ingênuo de fatoração, que procura por um divisor de $n$ entre 2 e $\lfloor \sqrt{n} \rfloor$, inclusive, consome tempo $\Theta(\beta^2 \sqrt{n}) = \Theta(\beta^2 2^{\beta/2})$ no pior caso, e portanto é exponencial em relação ao tamanho da entrada.

### C.5.1 O algoritmo de Euclides

Em diversos algoritmos de teoria dos números é essencial saber como calcular eficientemente o máximo divisor comum entre dois inteiros positivos. O teorema C.9 da Recursão de Euclides nos fornece imediatamente o seguinte algoritmo, conhecido como *algoritmo de Euclides*:

**Algoritmo** EUCLIDES-MDC $(a, b)$

1 se $b = 0$
2 então devolva $a$
3 senão devolva EUCLIDES-MDC $(b, a \bmod b)$

Vamos analisar o número de chamadas recursivas feitas pelo algoritmo ao executarmos EUCLIDES-MDC $(a, b)$. Podemos supor que $a > b \geq 0$. De fato, se

$b > a \geq 0$, então a chamada recursiva seguinte será EUCLIDES-MDC$(b, a)$. É claro também que, se $b = a > 0$, então EUCLIDES-MDC$(a, b)$ faz apenas uma chamada recursiva.

Defina os números de Fibonacci da seguinte maneira:

$$F_n := \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ F_{n-1} + F_{n-2} & \text{se } n > 1 \end{cases} \tag{C.17}$$

Referimos o leitor ao livro de Graham et al. [GKP94] para um estudo mais detalhado desta seqüência de números.

**Lema C.39.** *Se $a > b \geq 1$ e a execução de* EUCLIDES-MDC$(a, b)$ *faz $k \geq 1$ chamadas recursivas, então $a \geq F_{k+2}$ e $b \geq F_{k+1}$.*

*Demonstração.* Vamos provar o lema por indução em $k$. Para a base da indução, temos que $b \geq 1 = F_2$ e $a \geq 2 = F_3$ por hipótese.

Para o passo da indução, suponha que a execução de EUCLIDES-MDC$(a, b)$ faz $k > 1$ chamadas recursivas. A primeira chamada recursiva é EUCLIDES-MDC$(b, a \bmod b)$ que, por sua vez, realiza $k - 1$ chamadas recursivas. Pela hipótese de indução, temos então $b \geq F_{k+1}$ e $(a \bmod b) \geq F_k$. Assim, resta provarmos que $a \geq F_{k+2}$.

Como $a > b > 0$, então $\lfloor a/b \rfloor \geq 1$, de forma que

$$b + (a \bmod b) = b + (a - \lfloor a/b \rfloor b) \leq a.$$

Mas então $a \geq b + (a \bmod b) \geq F_{k+1} + F_k = F_{k+2}$, como queríamos. □

Segue imediatamente a seguinte delimitação para o número de chamadas recursivas:

**Teorema C.40 (Lamé).** *Se $a > b \geq 1$ e $b < F_{k+1}$ para algum inteiro $k \geq 1$, então* EUCLIDES-MDC$(a, b)$ *faz menos de $k$ chamadas recursivas.*

A delimitação apresentada pelo teorema C.40 é justa: basta verificar que, para qualquer $k \geq 1$, a execução de EUCLIDES-MDC$(F_{k+1}, F_k)$ faz exatamente $k - 1$ chamadas recursivas.

Não é difícil provar por indução que, para todo $n \geq 0$,

$$F_n = \frac{1}{\sqrt{5}}\left(\phi^n - \hat{\phi}^n\right), \tag{C.18}$$

onde $\phi := (1 + \sqrt{5})/2$ é a razão áurea e $\hat{\phi} := (1 - \sqrt{5})/2$. Assim, $F_n$ é aproximadamente $\phi^n/\sqrt{5}$. Combinando esta aproximação com o teorema C.40 de

Lamé, concluímos que o número de chamadas recursivas numa execução de Euclides-MDC $(a, b)$ é $O(\lg b)$. Supondo que $a$ e $b$ são inteiros de $\beta$ bits e que a divisão entre inteiros de $\beta$ bits consome $O(\beta^2)$ operações sobre bits, o total de operações sobre bits realizado por Euclides-MDC $(a, b)$ é de $O(\beta^3)$. Com um pouco mais de cuidado, pode-se provar que Euclides-MDC $(a, b)$ consome, na verdade, $O(\beta^2)$ operações sobre bits.

O gargalo do algoritmo de Euclides está no uso repetido de divisões entre inteiros de comprimento arbitrário. Se os inteiros $a$ e $b$ estiverem armazenados na base binária, como é usual, então podemos calcular $\mathrm{mdc}(a, b)$ utilizando apenas operações rápidas para a aritmética binária, como subtração, teste de paridade e divisão inteira por 2. O algoritmo resultante, conhecido como *binary gcd algorithm*, utiliza apenas $O(\lg a)$ operações binárias. Mais detalhes podem ser vistos no capítulo 31 do livro de Cormen et al. [CLRS01] e no livro de Knuth [Knu81].

O algoritmo de Euclides pode ser facilmente modificado para calcular inteiros $x$ e $y$ tais que $\mathrm{mdc}(a, b) = ax + by$, cuja existência é garantida pelo teorema C.7. É trivial verificar que o algoritmo a seguir, conhecido como *algoritmo estendido de Euclides*, realiza tal tarefa:

**Algoritmo** Euclides-Estendido-MDC $(a, b)$

se $b = 0$
  então devolva $(a, 1, 0)$
$(d', x', y') \leftarrow$ Euclides-Estendido-MDC $(b, a \bmod b)$
$(d, x, y) \leftarrow (d', y', x' - \lfloor a/b \rfloor y')$
devolva $(d, x, y)$

Os inteiros $x, y$ fornecidos pelo algoritmo estendido de Euclides certificam que a resposta devolvida pelo algoritmo está correta. De fato, é fácil verificar se $d = ax + by$, como o algoritmo garante. Se isso vale, então todo divisor de $a$ e de $b$ também divide $d$, de modo que $\mathrm{mdc}(a, b) \mid d$. Também é fácil verificar se $d$ é divisor comum de $a$ e $b$. Se isso também vale, é óbvio que $\mathrm{mdc}(a, b) = d$.

## C.5.2 Exponenciação modular

Assim como o cálculo do máximo divisor comum, a exponenciação modular é uma operação fundamental para diversos algoritmos de teoria dos números. Dados inteiros $a$ e $b$ não-negativos e $n$ positivo, queremos calcular eficientemente o valor $a^b \bmod n$. O algoritmo ingênuo que realiza $b$ multiplicações é obviamente exponencial em $\lg b$.

Uma idéia simples, mas poderosa, vem da utilização da representação binária do expoente $b$. Suponha que $b_k b_{k-1} \ldots b_0$ é a representação de $b$ em binário, onde $k := \lfloor \lg b \rfloor$ e, obviamente, $b_i \in \{0, 1\}$ para todo $i$. Em outras palavras, suponha que $b = \sum_{i=0}^{k} c_i$, onde $c_i := b_i 2^i$. Seja $B := \{i : b_i = 1\}$. Então

$$a^b = a^{\sum_{i=0}^{k} c_i} = \prod_{i=0}^{k} a^{c_i} = \prod_{i \in B} a^{2^i}. \tag{C.19}$$

A equação (C.19) nos fornece imediatamente o seguinte algoritmo, usualmente chamado de exponenciação por *repetição de quadrados (repeated squaring)*:

**Algoritmo** Repetição-de-Quadrados $(a, b, n)$

1 seja $b_k b_{k-1} \ldots b_0$ a representação binária de $b$
2 $x \leftarrow 1$
3 $y \leftarrow a$
4 para $i$ de 0 até $k$ faça
5     se $b_i = 1$
6         então $x \leftarrow xy \bmod n$
7     $y \leftarrow y^2 \bmod n$
8 devolva $x$

É fácil provar que o algoritmo calcula corretamente o valor $a^b \bmod n$, utilizando a equação (C.19) e o seguinte invariante: no início de cada iteração do laço das linhas 4–7, temos que $y = a^{2^i} \bmod n$.

Suponha que cada um dos inteiros $a, b$ e $n$ fornecidos como entrada têm $\beta$ bits. A representação binária de $b$, se não estiver imediatamente disponível, pode ser facilmente obtida através de $\beta$ divisões inteiras de $b$ por 2, o que consome tempo $O(\beta^2)$. Como $k = O(\beta)$, o laço das linhas 4–7 pode ser executado com $O(\beta^3)$ operações sobre bits, supondo que cada multiplicação e divisão pode ser realizada em tempo $O(\beta^2)$. Logo, o tempo de execução deste algoritmo é $O(\beta^3)$.

# Apêndice D

# Testes de primalidade e Criptografia RSA

Aplicamos agora os resultados vistos no capítulo sobre teoria dos números para compreendermos o teste de primalidade de Miller-Rabin e o sistema de criptografia RSA, que serve como motivação para o estudo do algoritmo de Shor.

## D.1 Os problemas da primalidade e da fatoração

O problema da primalidade consiste em, dado um inteiro $n > 1$, decidir se $n$ é primo. O problema da fatoração consiste em, dado um inteiro $n > 1$, encontrar sua fatoração única, como descrita no teorema C.12.

É evidente que o problema da primalidade está na classe **coNP**. De fato, se um inteiro $n > 1$ é composto, então $n$ tem um divisor não-trivial, isto é, existe um inteiro $1 < d < n$ que divide $n$. O fator $d$ é uma obstrução sucinta para a primalidade de $n$.

De forma não tão imediata, pode-se provar também que o problema da primalidade está na classe **NP**, partindo-se da seguinte caracterização de primos:

**Teorema D.1 (Teste de Lucas).** *Um inteiro $p > 1$ é primo se, e somente se, existe um inteiro $1 < r < p$ tal que $r^{p-1} \equiv 1 \pmod p$ e $r^{(p-1)/q} \not\equiv 1 \pmod p$ para todos os divisores primos $q$ de $p - 1$.*

A existência de certificados sucintos para primalidade é originalmente devida a Pratt [Pra75]:

**Teorema D.2 (Pratt).** *O problema da primalidade está na classe* **NP**.

Seria interessante, então, que os testes de primalidade, ou seja, os algoritmos que resolvem o problema da primalidade, nos fornecessem respostas afirmativas acompanhadas de certificados sucintos e respostas negativas acompanhadas de obstruções sucintas. Tal certificado sucinto não precisa, necessariamente, ser o mesmo certificado cuja existência foi provada por Pratt.

Da mesma forma, as obstruções devolvidas não precisam ser, necessariamente, fatores de um número composto $n$. Por exemplo, algumas das obstruções à primalidade devolvidas pelo Teste de Miller-Rabin, que vamos apresentar mais adiante, não oferecem nenhuma pista fácil sobre quais devem ser os fatores de $n$: não se sabe como se utilizar eficientemente a obstrução sucinta devolvida para se obter um fator de $n$. É claro que um teste polinomial de primalidade que devolve como obstrução um fator de $n$ pode ser utilizado para fatorar $n$ em tempo polinomial, ou seja, tal algoritmo estaria, na verdade, resolvendo o problema da fatoração.

Acredita-se que, no modelo clássico, fatorar um número seja mais difícil do que decidir se ele é primo ou não. O sistema de criptografia de chave pública mais amplamente utilizado atualmente, o RSA, baseia-se exatamente nessa suposição e no conhecimento de testes polinomiais de primalidade. Os testes são empregados na busca de primos grandes, necessários para a geração de chaves. Sem testes eficientes de primalidade, seria muito complicado criar chaves seguras. Já a dificuldade de criptanálise do RSA baseia-se justamente na suposição de que é difícil fatorar números grandes.

Atualmente, o algoritmo probabilístico de fatoração mais rápido é o *general number field sieve*, originalmente desenvolvido por Pollard [Pol93] e Lenstra *et al.* [LLMP93] e aperfeiçoado por Coppersmith [Cop93] e Buhler *et al.* [BLP93]. É difícil obter uma análise de tempo rigorosa para esse algoritmo, mas, partindo de suposições razoáveis, pode-se mostrar que o algoritmo consome tempo proporcional a

$$\exp\left(c\,(\ln n)^{1/3}(\ln\ln n)^{2/3}\right),$$

onde $c := \frac{1}{3}\left(92+26\sqrt{13}\right)^{1/3} \cong 1{,}902$.

Outro algoritmo probabilístico de fatoração é o método das curvas elípticas, desenvolvido por Lenstra [Len87]. De acordo com Pomerance [Pom96], esse algoritmo é mais lento que o *general number field sieve* apenas para uma pequena fração dos números compostos. Dado um inteiro $n$ composto, esse método encontra um fator primo pequeno $p$ de $n$ em tempo estimado proporcional a

$$\exp\left(d\,(\ln p)^{1/2}(\ln\ln p)^{1/2}\right),$$

onde $d \cong \sqrt{2}$. Note que o tempo de execução de ambos algoritmos é superpolinomial em $\log n$.

A história é bastante diferente para testes de primalidade.

O teste de Miller-Rabin [Mil75, Mil76, Rab80] é um algoritmo probabilístico polinomial com erro unilateral limitado. Seguindo a terminologia de Papadimitriou [Pap94], esse teste é um algoritmo polinomial de Monte Carlo para dizer se um inteiro é composto, o que significa que, se o teste afirma que um número $n$ é composto, então $n$ é composto com certeza. Já se o teste responde que $n$ é primo, existe a possibilidade de ele estar errado, isto é, é possível que $n$ seja composto. Porém, a probabilidade de ocorrência deste erro é estritamente menor que $1/2$. Isso estabelece que o problema da primalidade está na classe **coRP**. Além disso, o teste de Miller-Rabin, ao afirmar que um dado número é composto, nos devolve uma obstrução sucinta à sua primalidade.

Existem algoritmos polinomiais de Monte Carlo para dizer se um inteiro é primo, isto é, o erro unilateral limitado destes testes é complementar ao erro do teste de Miller-Rabin. Ou seja, se o teste afirma que um número $n$ é primo, então certamente isso é verdade. Já se o teste responde que $n$ é composto, a probabilidade de ele estar errado é estritamente menor que $1/2$. Entre esses testes está o algoritmo de Adleman e Huang [AH92], obtido a partir de modificações no algoritmo de Goldwasser e Kilian [GK86]. O algoritmo de Adleman e Huang, ao responder que um dado número é primo, nos devolve um certificado sucinto de primalidade.

Estabelecemos assim que o problema da primalidade está na classe **RP**. Mas então o problema da primalidade está na classe $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$, de modo que ele admite um algoritmo de Las Vegas, seguindo novamente a terminologia de Papadimitriou [Pap94]. Isto é, ao repetirmos $k$ execuções alternadas dos algoritmos de Miller-Rabin e de Adleman e Huang, a probabilidade de nunca obtermos uma resposta definitiva é menor que $2^{-k}$. Estamos chamando de resposta definitiva as respostas dadas com certeza por cada algoritmo: no caso do teste de Miller-Rabin, a resposta definitiva é a afirmação de que o número é composto e, no caso do teste de Adleman e Huang, a afirmação de que o número é primo. Observe que, com este algoritmo de Las Vegas, não só a probabilidade de não obtermos respostas definitivas é arbitrariamente pequena, mas também obtemos certificados e obstruções sucintas de primalidade, o que é muito conveniente.

Essa linha de pesquisa culminou com o AKS, um algoritmo determinístico polinomial para o problema da primalidade, desenvolvido por Agrawal, Kayal e Saxena [AKS02a, AKS02b]. Provou-se assim que o problema da primalidade está na classe **P**, de modo que todos os resultados anteriores tornaram-se corolários deste.

Um histórico mais detalhado sobre o desenvolvimento de testes de primalidade pode ser visto na seção introdutória dos artigos de Agrawal, Kayal e Saxena [AKS02a, AKS02b].

## D.2 O Teste de Miller-Rabin

O teste de primalidade de Miller-Rabin [Mil75, Mil76, Rab80] é um algoritmo probabilístico extremamente simples e rápido. Porém, sua análise não é tão trivial. Utilizaremos quase todo o conteúdo das seções anteriores para sua exposição.

Dado um inteiro $n > 1$ par, é óbvio que $n$ é primo se e só se $n = 2$. Todos os outros pares positivos têm o divisor 2 como obstrução sucinta de primalidade. Nesta seção, vamos nos restringir ao problema da primalidade apenas para inteiros ímpares estritamente maiores que 1.

Uma primeira idéia para um teste de primalidade é dada pela contrapositiva do teorema C.25 de Fermat: se $a^{n-1} \not\equiv 1 \pmod{n}$ para algum $0 < a < n$, então $n$ é composto. De fato, se $n$ fosse primo, então $a^{n-1} \equiv 1 \pmod{n}$ para todo $0 < a < n$, pois todo $0 < a < n$ é relativamente primo a $n$. Podemos dizer que $a$ é uma *testemunha* do fato de $n$ ser composto.

Segundo Coutinho [Cou00], Leibniz utilizou esta idéia como critério de primalidade. Dado um inteiro $n > 1$ ímpar, se $2^{n-1} \not\equiv 1 \pmod{n}$, o teste de Leibniz responde que $n$ é composto e que 2 é uma testemunha deste fato. Caso contrário, ele afirma que $n$ é primo.

Observe que esse teste responde que $341 = 11 \times 31$ é primo. Existem diversos outros inteiros compostos para os quais o teste de Leibniz dá a resposta errada. Isso nos sugere a seguinte definição:

**Definição D.3.** *Sejam $n > 1$ um número composto e $0 < a < n$. Se*

$$a^{n-1} \equiv 1 \pmod{n}, \tag{D.1}$$

*então $n$ é* pseudoprimo para a base *$a$.*

O teste de Leibniz sempre dá a resposta errada para pseudoprimos para a base 2, pois afirma que eles são primos.

Infelizmente, não basta estendermos o teste de Leibniz para que a equivalência (D.1) seja testada para alguma outra base, por exemplo, $a = 3$. A razão disso é que existem inteiros $n$ que são pseudoprimos para todas as bases relativamente primas a $n$. Tais inteiros são chamados *números de Carmichael* [Car12]. O menor número de Carmichael é $561 = 3 \times 11 \times 17$. Alford *et al.* [AGP94] provaram que existem infinitos números de Carmichael.

O teste de Miller-Rabin contorna essas falhas do teste de Leibniz, combinando o uso do teorema C.25 e do corolário C.37 com aleatorização para obter um teste probabilístico eficiente de primalidade, com erro limitado unilateral.

Resumidamente, a base $a$ da equivalência (C.9) é escolhida aleatoriamente e, durante o cálculo da exponenciação modular $a^{n-1} \bmod n$, procuramos por raízes quadradas não-triviais de 1, módulo $n$. O teste responde que $n$ é composto apenas se encontrar tais raízes ou se $a$ e $n$ não satisfizerem a equação (C.9) do teorema C.25 de Fermat.

Detalhamos a seguir o pseudocódigo para o teste de Miller-Rabin:

**Algoritmo** Miller-Rabin ($n$)

```
01    escolha um inteiro 0 < a < n aleatoriamente,
                com distribuição uniforme
02    sejam u um ímpar e t ≥ 1 inteiros tais que n − 1 = 2^t u
03    x_0 ← a^u mod n
04    para i de 1 a t faça
05        x_i ← x_{i−1}^2 mod n
06        se x_i = 1 e x_{i−1} ≠ 1 e x_{i−1} ≠ n − 1
07            então devolva "composto", acompanhado de a
08    se x_t ≠ 1
09        então devolva "composto", acompanhado de a
10        senão devolva "primo"
```

Se a linha 3 for executada através do método de repetição de quadrados, então este algoritmo é apenas uma leve modificação do procedimento Repetição-de-Quadrados, apresentado anteriormente, e seu tempo de execução é $O(\beta^3)$, supondo que $n$ é um inteiro de $\beta$ bits.

Caso o algoritmo devolva "composto" através da linha 7, então ele encontrou uma raiz quadrada não-trivial de 1, módulo $n$ e, de acordo com o corolário C.37, o inteiro $n$ é composto. Essa raiz é, no caso, $a^b \bmod n$, onde $b := 2^{i-1}u$. Já se a resposta "composto" for proveniente da linha 9, então $a^{n-1} \not\equiv 1 \pmod{n}$ e, de acordo com a contrapositiva do teorema C.25 de Fermat, o número $n$ é composto. Em ambos os casos, vamos chamar a base $a$ de *testemunha de que $n$ é composto*.

Se o algoritmo não encontrou evidências de que $n$ é composto, então ele responde que $n$ é primo, apesar de não ter certeza. Precisamos agora limitar a probabilidade deste erro, proveniente de "azar" na escolha da base $a$.

Faremos uso do seguinte lema:

**Lema D.4.** *Seja $n$ um número de Carmichael. Então $n$ não é potência de primo.*

*Demonstração.* Seja $n$ um número de Carmichael. Então, para todo $x \in \mathbb{Z}_n^*$, temos

$$x^{n-1} \equiv 1 \pmod{n}. \tag{D.2}$$

Suponha que $n := p^k$, onde $p$ é um primo e $k > 1$. Como $n$ é ímpar, $p$ também deve ser ímpar. Pelo teorema C.28, o grupo $\mathbb{Z}_n^*$ é cíclico, ou seja, $\mathbb{Z}_n^*$ contém um gerador $g$ tal que $\text{ord}_n(g) = |\mathbb{Z}_n^*| = \phi(n)$. Pelo teorema C.32, $\phi(n) = p^k\,(1-1/p) = (p-1)\,p^{k-1}$. Pela equação (D.2), temos $g^{n-1} \equiv 1 \pmod{n}$, ou seja, $g^{n-1} \equiv g^0 \pmod{n}$. Então, pelo teorema C.35 do logaritmo discreto, temos $n - 1 \equiv 0 \pmod{\phi(n)}$, isto é,

$$(p-1)\,p^{k-1} \,\big|\, (p^k - 1).$$

Como $k > 1$, então $p$ divide $(p-1)\,p^{k-1}$. Mas $p$ não divide $p^k - 1$. Isso é um absurdo. Segue que $n$ não é potência de primo. $\square$

A demonstração do teorema a seguir segue de perto a prova apresentada no livro de Cormen *et al.* [CLRS01].

**Teorema D.5.** *Seja $n$ um composto ímpar. Então o número de testemunhas de que $n$ é composto é estritamente maior que $(n-1)/2$.*

*Demonstração.* Sejam $n$ um composto ímpar e $0 < a < n$ um inteiro. Se uma chamada a Miller-Rabin $(n)$ responder que $n$ é primo ao escolher $a$ como base na linha 1, dizemos que $a$ é uma *não-testemunha* de que $n$ é composto. Vamos mostrar que o número de não-testemunhas de que $n$ é composto é estritamente menor que $(n-1)/2$, de onde o teorema segue imediatamente.

É fácil ver que toda não-testemunha está em $\mathbb{Z}_n^*$. De fato, suponha que $a$ é uma não-testemunha de que $n$ é composto. Então $x_t = 1$ na linha 8 do algoritmo, ou seja, $a^{n-1} \equiv 1 \pmod{n}$. Logo, a equação $ax \equiv 1 \pmod{n}$ admite uma solução, a saber, $x = a^{n-2}$. Pelo corolário C.34, temos que $\text{mdc}(a, n) \mid 1$ e, portanto, $\text{mdc}(a, n) = 1$. Segue que toda não-testemunha está em $\mathbb{Z}_n^*$.

Vamos mostrar que toda não-testemunha está num subgrupo próprio $B$ de $\mathbb{Z}_n^*$. Pelo corolário C.18, teremos então $|B| \leq |\mathbb{Z}_n^*|/2 = \phi(n)/2$. Como $n$ é composto, então $\phi(n) < n - 1$, de modo que teremos $|B| < (n-1)/2$ e a demonstração estará concluída.

Começamos tratando o caso em que $n$ não é um número de Carmichael, ou seja, existe $x \in \mathbb{Z}_n^*$ tal que $x^{n-1} \not\equiv 1 \pmod{n}$. Seja $B := \{b \in \mathbb{Z}_n^* : b^{n-1} \equiv 1 \pmod{n}\}$. Como $1 \in B$, o conjunto $B$ não é vazio. Além disso, é evidente que $B$ é fechado sob multiplicação módulo $n$. Então $B$ é um subgrupo de $\mathbb{Z}_n^*$. Observe que toda não-testemunha está em $B$, pois toda não-testemunha $a$ satisfaz $a^{n-1} \equiv 1 \pmod{n}$. Como $x \in \mathbb{Z}_n^* \setminus B$, então $B$ é um subgrupo próprio de $\mathbb{Z}_n^*$, como queríamos.

Suponha, de agora em diante, que $n$ é um número de Carmichael.

Sejam $u$ um ímpar e $t \geq 1$ inteiros tais que $n - 1 = 2^t u$. Dizemos que um par $(v, j)$ de inteiros é *aceitável* se $v \in \mathbb{Z}_n^*$, $0 \leq j \leq t$ e

$$v^{2^j u} \equiv -1 \pmod{n}.$$

Pares aceitáveis certamente existem já que $u$ é ímpar: tome, por exemplo, $v = n - 1$ e $j = 0$. Escolha o maior $j$ possível tal que existe um par $(v, j)$ aceitável e fixe $v$ de modo que $(v, j)$ seja aceitável. Seja

$$B = \left\{x \in \mathbb{Z}_n^* : x^{2^j u} \equiv \pm 1 \pmod{n}\right\}.$$

É fácil ver que $B$ é fechado sob multiplicação módulo $n$, e portanto é um subgrupo de $\mathbb{Z}_n^*$.

Toda não-testemunha está em $B$. De fato, considere a seqüência

$$X := \langle x_0, x_1, x_2, \ldots, x_t \rangle := \left\langle a^u, a^{2u}, a^{2^2 u}, \ldots, a^{2^t} u \right\rangle$$

de inteiros módulo $n$ calculada pelo algoritmo de Miller-Rabin. As possíveis "formas" de $X$ são:

- $X = \langle \ldots, d \rangle$, com $d \not\equiv 1 \pmod{n}$. Então $a^{n-1} \not\equiv 1 \pmod{n}$ e o algoritmo responde, na linha 9, que $n$ é composto.

- $X = \langle 1, \ldots, 1 \rangle$. Neste caso, a base $a$ escolhida na linha 1 não é testemunha de que $n$ é composto e o algoritmo responde que $n$ é primo.

- $X = \langle \ldots, -1, 1, \ldots, 1 \rangle$. Como a seqüência termina em 1 e a última entrada diferente de 1 é $-1$, a base $a$ escolhida na linha 1 não é testemunha de que $n$ é composto. O algoritmo responde que $n$ é primo.

- $X = \langle \ldots, d, 1, \ldots, 1 \rangle$, com $d \not\equiv \pm 1 \pmod{n}$. A seqüência termina em 1, mas o último valor diferente de 1 é $d \not\equiv \pm 1 \pmod{n}$, e portanto foi encontrada uma raiz quadrada não-trivial de 1, módulo $n$. O algoritmo responde, na linha 7, que $n$ é composto.

Note então que, se $a$ é uma não-testemunha, então a seqüência $X$ produzida para ela é da forma $X = \langle 1, \ldots, 1 \rangle$ ou $X = \langle \ldots, -1, 1, \ldots, 1 \rangle$. Se a seqüência é toda de 1's, então é óbvio que $a \in B$. Já se $X = \langle \ldots, -1, 1, \ldots, 1 \rangle$, então a posição em que $-1$ ocorre não é maior que $j$, pela maximalidade de $j$. Assim, toda não-testemunha está em $B$.

Vamos mostrar agora que $B$ é um subgrupo próprio de $\mathbb{Z}_n^*$. Para tanto, vamos provar a existência de um $w \in \mathbb{Z}_n^* \setminus B$ através dos corolários do teorema C.29 do resto chinês.

Pelo lema D.4, $n$ não é potência de primo, já que $n$ é um número de Carmichael. Então podemos escrever $n$ como um produto $n = n_1 n_2$, onde $n_1$ e $n_2$ são ímpares primos entre si, ambos maiores que 1.

Como $v^{2^j u} \equiv -1 \pmod{n}$, o corolário C.31 nos garante que

$$v^{2^j u} \equiv -1 \pmod{n_1}. \tag{D.3}$$

Considere o sistema de equações

$$\begin{aligned} y &\equiv v \pmod{n_1} \\ y &\equiv 1 \pmod{n_2}, \end{aligned} \tag{D.4}$$

com incógnita $y$. Seja $w$ uma solução do sistema (D.4), cuja existência é garantida pelo corolário C.30. Como $w \equiv v \pmod{n_1}$, segue da equação (D.3) que

$$w^{2^j u} \equiv -1 \pmod{n_1} \tag{D.5}$$

$$w^{2^j u} \equiv +1 \pmod{n_2}. \tag{D.6}$$

Pelo corolário C.31, $w^{2^j u} \not\equiv 1 \pmod{n_1}$ implica que $w^{2^j u} \not\equiv 1 \pmod{n}$ e $w^{2^j u} \not\equiv -1 \pmod{n_2}$ implica que $w^{2^j u} \not\equiv -1 \pmod{n}$. Então $w^{2^j u} \not\equiv \pm 1 \pmod{n}$, de modo que $w \notin B$.

Resta apenas mostrarmos que $w \in \mathbb{Z}_n^*$. Como $v \in \mathbb{Z}_n^*$, então $\mathrm{mdc}(v, n) = 1$ e, portanto, $\mathrm{mdc}(v, n_1) = 1$. Dado que $w \equiv v \pmod{n_1}$, então $n_1 \mid w - v$. Seja $d > 1$ um divisor de $n_1$, de modo que $d \mid w - v$. Como $d \nmid v$, então não podemos ter $d \mid w$. Segue que $\mathrm{mdc}(w, n_1) = 1$.

A equivalência $w \equiv 1 \pmod{n_2}$ implica que $\mathrm{mdc}(w, n_2) = 1$. Como $w$ não tem divisores não-triviais em comum nem com $n_1$ e nem com $n_2$, então $\mathrm{mdc}(w, n_1 n_2) = 1$, ou seja, $w \in \mathbb{Z}_n^*$, como queríamos. □

Segue imediatamente do teorema D.5 que a probabilidade de o teste de Miller-Rabin responder que $n$ é primo quando, na verdade, $n$ é composto, é estritamente menor que $1/2$.

Monier [Mon80] provou que, para todo $n$ composto ímpar, existem no máximo $(n-1)/4$ bases $0 < a < n$ que não são testemunhas de que $n$ é composto, e esta delimitação é justa. Isso diminui consideravelmente a probabilidade de erro do teste de Miller-Rabin, comparado à delimitação obtida pelo teorema D.5.

Seja $n$ um composto e suponha que uma chamada a MILLER-RABIN $(n)$ respondeu que $n$ é composto através da linha 7, acompanhado de uma base $a$, testemunha de que $n$ é composto. É muito fácil obter uma raiz quadrada não-trivial $x$ de 1, módulo $n$, a partir de $a$: basta seguir as linhas 2–7 do mesmo algoritmo. Com esta raiz em mãos, é muito fácil obter um fator de $n$, utilizando o teorema C.38.

Agora suponha que a testemunha $a$ devolvida é proveniente da linha 9. Neste caso, não temos garantias de que é possível obter um fator de $n$ a partir de $a$. A idéia mais óbvia é verificar se $\mathrm{mdc}(a, n) > 1$, mas claramente não temos como limitar inferiormente a probabilidade desse evento.

Concluímos que não é uma tarefa óbvia a utilização do teste de Miller-Rabin e das obstruções sucintas por ele devolvidas para se resolver eficientemente o problema da fatoração.

## D.3 O sistema de criptografia RSA

Suponha que Alice quer enviar para Beto uma mensagem privada através de um canal de comunicação inseguro. Isto é, suponha que existe uma intrusa mal-intencionada Eva capaz de interceptar mensagens enviadas através do canal.

Considere a seguinte solução ingênua para este problema. Alice e Beto combinam previamente o uso de uma chave, que será mantida em segredo pelos dois. Quando Alice deseja enviar uma mensagem privada para Beto, ela utiliza a chave secreta para codificar a mensagem, de modo que o texto produzido seja ininteligível por um interceptador que desconheça a chave, como Eva. Beto, ao receber a mensagem codificada, utiliza a chave para decodificar a mensagem, obtendo o texto original escrito por Alice.

Um jeito simples de se codificar e decodificar uma mensagem utilizando uma chave é utilizar esta como um inteiro indicando um deslocamento no alfabeto. Assim, se a chave for 3, toda letra 'a' da mensagem vira 'd' na mensagem codificada, todo 'b' vira 'e', e assim por diante. Vale lembrar que este método de criptografia, utilizado com sucesso por muitos séculos, é muito inseguro, pois é facilmente quebrado.

A principal falha desta abordagem é que ela exige que se combine previamente uma chave secreta. Isso impossibilita seu uso em diversas situações em que as entidades que desejam se comunicar não se conhecem, como por exemplo no comércio eletrônico.

Para contornar essa falha, Diffie e Hellman [DH76] introduziram o conceito de criptografia de chave pública. Num sistema desse tipo, o envio da mensagem transcorreria da seguinte forma. Beto teria duas chaves: uma pública e outra secreta. A chave pública, diferente da secreta, seria acessível a todos. Alice, antes de enviar a mensagem, utilizaria a chave pública de Beto para codificá-la. Apenas Beto, através de sua chave secreta, é capaz de decodificar a mensagem codificada enviada por Alice.

Vamos descrever o sistema mais formalmente. Seja $\mathcal{M}$ o conjunto de mensagens possíveis. Por exemplo, $\mathcal{M}$ pode ser o conjunto de todas as cadeias de caracteres com comprimento finito. Seja $P_B$ a chave pública de Beto e $S_B$ sua chave secreta. Tanto $P_B$ quanto $S_B$ definem funções bijetoras em $\mathcal{M}$. Por isso, vamos chamar essas funções também de $P_B$ e $S_B$. Dada uma mensagem $M \in \mathcal{M}$, a mensagem codificada é dada por $P_B(M)$, isto é, utiliza-se a chave pública para a codificação. Para decodificar tal mensagem, queremos que $S_B(P_B(M)) = M$, isto é, a decodificação utiliza a chave secreta para obter a mensagem original. O detalhe essencial é que $P_B$ e $S_B$ devem ser escolhidos de forma que seja impraticável descobrir $S_B$ apenas a partir de $P_B$ (lembre-se que qualquer um tem acesso à chave pública $P_B$).

Um sistema com essas características pode ser utilizado também para criar assinaturas digitais. Por exemplo, suponha que Alice deseja assinar a mensagem a ser enviada para Beto, de forma que ninguém seja capaz de falsificar sua assinatura e que Beto seja capaz de se certificar de que Alice foi de fato o remetente.

Seja $P_A$ a chave pública de Alice e $S_A$ sua chave secreta. Tais chaves definem funções bijetoras $P_A$ e $S_A$ em $\mathcal{M}$. Como devemos ter $S_A(P_A(M)) = M$, então as funções $P_A$ e $S_A$ são uma a inversa da outra, de modo que temos também $P_A(S_A(M)) = M$. Como apenas Alice tem acesso à chave $S_A$, ela pode utilizar $\sigma := S_A(M)$ como assinatura da mensagem $M$. Ela pode enviar o par $(M, \sigma)$ a Beto que, por sua vez, pode verificar se $P_A(\sigma) = M$ para se certificar de que Alice foi o remetente. Desta forma, apenas Alice é capaz de gerar sua assinatura. Além disso, se de alguma forma Eva conseguiu alterar a mensagem antes de Beto recebê-la, este será capaz de detectar a invalidade da mensagem.

É claro que as operações de assinatura e de codificação de uma mensagem podem ser compostas para se obter uma mensagem assinada por Alice que apenas Beto possa ler.

Rivest, Shamir e Adleman [RSA78] construíram um sistema de criptografia de chave pública, conhecido como RSA, que se baseia na relativa facilidade de se encontrar números primos grandes e na dificuldade de se fatorar inteiros.

Vamos mostrar como Beto faria para criar suas chaves no RSA. Primeiro ele escolhe aleatoriamente dois primos grandes $p$ e $q$. Para tanto, basta sortear um inteiro ímpar grande e executar algum teste de primalidade, como o AKS, ou um número suficiente de execuções do teste de Miller-Rabin. Queremos delimitar inferiormente a probabilidade de o número sorteado ser primo. Podemos utilizar o seguinte resultado, conhecido como teorema dos números primos.

**Teorema D.6.** *Seja $\pi(n) : \mathbb{N} \longrightarrow \mathbb{N}$ a função definida por*

$$\pi(n) := \big|\{p : p < n \text{ e } p \text{ primo}\}\big|. \tag{D.7}$$

*Então $\pi(n) \sim n/\ln n$.*

Portanto a probabilidade de que o inteiro sorteado seja primo se aproxima de $1/\ln n$ para $n$ grande. Assim, o número esperado de sorteios necessários para se obter um primo é $\ln n$, o que é razoável.

Com os primos $p$ e $q$ em mãos, Beto calcula o produto $n := pq$. Agora ele escolhe um ímpar $e$ pequeno que seja relativamente primo a $\phi(n) = (p-1)(q-1)$, onde utilizamos o teorema C.32 para calcular $\phi(n)$. Como $\text{mdc}\big(e, \phi(n)\big) = 1$, então $e$ tem um inverso multiplicativo módulo $\phi(n)$, que pode ser calculado através do algoritmo estendido de Euclides, descrito na seção C.5.1. Seja $d$ tal inverso.

A chave pública de Beto será o par $(e, n)$. A chave secreta é $(d, n)$. Vamos considerar que o conjunto das possíveis mensagens é $\mathcal{M} := \mathbb{Z}_n$. A função associada com a chave pública $(e, n)$ é

$$P_B(M) := M^e \bmod n. \tag{D.8}$$

A função associada à chave secreta é:

$$S_B(M) := M^d \bmod n. \tag{D.9}$$

Ambas as funções são facilmente calculadas utilizando o algoritmo de exponenciação modular, descrito na seção C.5.2. De fato, seja $\beta := \lfloor \lg n \rfloor + 1$ o número de bits de $n$. Então todas as operações descritas acima podem ser realizadas com $O(\beta^3)$ operações sobre bits.

Temos que mostrar que tais funções satisfazem as propriedades desejadas, de acordo com nossa especificação de um sistema de criptografia de chave pública. Começamos observando que as funções $P_B$ e $S_B$, comutam, isto é, vale que $P_B(S_B(M)) = S_B(P_B(M))$ para todo $M \in \mathcal{M} = \mathbb{Z}_n$.

Seja $M \in \mathcal{M} = \mathbb{Z}_n$. Temos $S_B(P_B(M)) \equiv M^{ed} \pmod{n}$. Como temos $ed \equiv 1 \pmod{\phi(n)}$, então $ed = 1 + k\phi(n) = 1 + k(p-1)(q-1)$ para algum inteiro $k$. Se $M \not\equiv 0 \pmod{p}$, então

$$\begin{aligned} M^{ed} &\equiv M\big(M^{p-1}\big)^{k(q-1)} && \pmod{p} \\ &\equiv M(1)^{k(q-1)} && \pmod{p} \\ &\equiv M && \pmod{p}, \end{aligned}$$

onde utilizamos o teorema C.25 de Fermat para passar da primeira para a segunda linha. A relação $M^{ed} \equiv M \pmod{p}$ também vale se $M \equiv 0 \pmod{p}$. Analogamente, temos $M^{ed} \equiv M \pmod{q}$ para todo $M \in \mathbb{Z}_n$. Então, pelo corolário C.31 ao teorema do resto chinês, temos $M^{ed} \equiv M \pmod{n}$ para todo $M \in \mathbb{Z}_n$, como queríamos.

Observe que, se conseguirmos fatorar $n = pq$, então será trivial calcular $\phi(n) = (p-1)(q-1)$, bem como $d$, o inverso multiplicativo de $e$, módulo $\phi(n)$. Lembre-se que $e$ faz parte da chave pública de Beto, e que $d$ é parte da chave secreta. Assim, um algoritmo eficiente de fatoração pode ser utilizado na quebra do RSA.

# Apêndice E

# Circuitos clássicos

No modelo clássico de computação, vimos três tipos distintos de máquina de Turing. Especialmente importante é a MTD, que formaliza o modelo teórico que fundamenta os computadores atuais, que têm comportamentos governados pelas leis da mecânica clássica.

Como conseqüência, os dispositivos computacionais modernos têm uma série de restrições em suas primitivas básicas, afetando assim o poder computacional do modelo.

Porém, como sabemos que o universo segue as leis da mecânica quântica, é natural que nos perguntemos se é possível construir dispositivos computacionais baseados nas propriedades da mecânica quântica. Foi a partir dessa perspectiva que surgiu a Máquina de Turing Quântica.

Sucintamente, podemos dizer que a MTQ é uma MTP com amplitudes de probabilidade complexas. Além disso, tais máquinas admitem superposições de configurações e interferência, o que sugere que o modelo quântico tem possivelmente maior poder computacional.

Porém, a necessidade do uso de operações unitárias devido a "imposições" da mecânica quântica acaba levantando a seguinte questão: as evoluções unitárias, obrigatórias na mecânica quântica, restringem a classe de problemas eficientemente computáveis?

Para tentar esclarecer algumas dessas questões, analisamos o funcionamento dos circuitos clássicos. Em especial, mostramos que as operações realizadas no modelo clássico podem ser feitas com portas reversíveis, provando assim que o modelo quântico é pelo menos tão poderoso quanto o clássico.

Nesse capítulo, nosso objetivo é explicar o funcionamento das portas lógicas nos computadores atuais. Inicialmente, apresentamos o conceito de porta universal na seção E.1. Em seguida, na seção E.2, vamos mostrar como podemos analisar a complexidade de um determinado algoritmo através do circuito usado para implementá-lo. Por fim, veremos na seção E.3 que é possível construir uma máquina clássica que realiza computações reversíveis.

## E.1 Portas universais

Podemos dizer que a computação clássica envolve o cálculo de funções. Cada uma dessas funções pode ser vista, sem perda de generalidade, como uma função que tem como domínio o conjunto das palavras formadas pelos símbolos 0 e 1, e como imagem os números 0 e 1. Como os elementos da imagem podem representar rejeição e aceitação (de uma palavra em uma linguagem, por exemplo), essas funções serão denominadas *funções de decisão*.

Denote por $n$ o comprimento de uma palavra dada como entrada para uma função. Suponha que tal função tem como entrada apenas palavras de comprimento $n$. Como o número de entradas possíveis é $2^n$ e o número de respostas para cada entrada é 2, o número de funções de decisão desse tipo é $2^{2^n}$.

Todas as funções de decisão podem ser decompostas em seqüências de operações lógicas. Estamos interessados num conjunto de operações lógicas capaz de representar todas as essas funções.

Além disso, para que a construção de um dispositivo capaz de computar qualquer função de decisão seja viável, é interessante utilizar um conjunto de operadores razoavelmente sucinto. A inserção de um operador lógico para cada função seria claramente inviável para valores muito grandes de $n$.

Um resultado conhecido da álgebra booleana, que pode ser visto no livro de Feynman [Fey96], mostra que qualquer função de decisão pode ser construída com as portas AND e NOT. Claramente, qualquer combinação de portas capaz de simular essas duas tem o mesmo poder. Dizemos que um conjunto de portas com essa propriedade é um *conjunto completo de operadores*.

Vamos descrever uma *porta universal*, que nada mais é do que uma porta lógica que compõe sozinha um conjunto completo de operadores. Vale destacar, porém, que a obtenção de tal porta pode ser mero preciosismo em alguns casos, pois os custos envolvidos no uso de duas portas não é necessariamente maior que o do uso de uma única porta.

Como as portas AND e NOT são suficientes para representar qualquer função de decisão, se tivermos alguma porta capaz de simular essas duas então essa porta será universal. Uma porta que serve para tal propósito é a porta NAND, que é a negação da porta AND. É fácil construir a porta NOT a partir dela (basta colocar o sinal que deseja-se negar nas entradas da porta para que a negação seja obtida), e a partir da negação podemos obter a porta AND. Logo, toda função de decisão pode ser representada por algum circuito formado exclusivamente por portas NAND.

Por fim, vale destacar que a porta NAND, apesar de universal, não é reversível. Quando $n > 1$, se a saída devolvida for 1 não podemos determinar a entrada recebida pela porta. Na computação clássica esse fato é de menor relevância, mas na computação quântica isso é importante, pois todas as computações envolvidas devem ser reversíveis. Por isso, vamos comentar adiante portas universais reversíveis para o modelo clássico.

## E.2 Complexidade de circuitos

Conforme já dissemos anteriormente, as funções de decisão podem ser vistas como codificações de problemas de decisão. Um problema de decisão é aquele que, para toda entrada, tem como resposta 1 ou 0.

No estudo da complexidade, estamos interessados em saber qual é a dificuldade de um determinado problema. A dificuldade está diretamente relacionada aos recursos que uma solução vai consumir.

Quando desenvolvemos uma solução para um problema utilizando circuitos, é razoável considerar o tamanho dos circuitos montados, que é o número de portas lógicas, por exemplo, e o tempo consumido para que uma determinada entrada seja computada. O tempo consumido por um circuito está relacionado ao que chamamos de *profundidade do circuito*, que é o maior número de passos que um circuito que admite paralelismo deve executar para computar o valor de uma entrada. Já a *largura de um circuito* é o maior número de operações lógicas executadas simultaneamente.

Naturalmente, cada problema possui infinitas elaborações possíveis de circuitos que o resolvem (ou decidem). No estudo da complexidade, porém, estamos interessados em circuitos de fácil adaptabilidade. Ou seja, dado um circuito que resolve o problema para entradas de tamanho $n$, deve ser fácil montar um circuito para entradas de tamanho $n + 1$. Em geral, a facilidade está relacionada à existência de um algoritmo que consome tempo polinomial para realizar a tarefa.

Analisando o crescimento do tamanho dos circuitos em uma família de circuitos que resolvem um mesmo problema, podemos definir sua complexidade. Se o crescimento desses circuitos é limitado por algum polinômio, então dizemos que o problema que estamos tratando é fácil, pois ele admite uma solução polinomial.

Como podemos simular em uma máquina de Turing qualquer circuito em tempo polinomial no tamanho do circuito, concluímos que essa noção de facilidade coincide com a oriunda das máquinas de Turing. Ou seja, a classe de problemas que podem ser resolvidos por circuitos que crescem polinomialmente de acordo com o tamanho da entrada é a classe **P**, já discutida anteriormente.

Infelizmente, nem todos os problemas podem ser resolvidos por meio de circuitos de tamanho polinomial. Porém, existem problemas desse tipo que são interessantes. Em especial, temos problemas para os quais não se conhece solução fácil, mas cujas soluções podem ser verificadas por um circuito (ou algoritmo) polinomial. Os problemas com essas características pertencem à classe **NP**, também já discutida.

Dessa forma, estabelecemos a relação entre as classes de complexidade do modelo clássico e os circuitos que são montados para resolver os problemas.

## E.3 Computação reversível

Conforme afirmamos anteriormente, estamos interessados em construir operações reversíveis no modelo clássico. Para isso, precisaremos de mais duas portas para construir as funções desejadas. Denominaremos estas portas FANOUT e EXCHANGE. A primeira servirá para gerar dois "fios" com o mesmo pulso (ou bit) que o original. Já a segunda servirá para trocar os bits entre dois fios.

Conforme dissemos anteriormente, a porta NAND não é reversível. Da mesma forma, outras portas, como AND, OR, XOR e outras, também não são. Como queremos construir circuitos reversíveis, não será possível utilizar tais portas. Um exemplo de porta reversível é a porta NOT. Ao analisar a saída gerada por ela, sabemos qual foi o sinal recebido na entrada.

Para construir o conjunto de portas necessárias para representar todas as funções de decisão, vamos utilizar, junto com a porta NOT (N), as portas CONTROLLED NOT (CN) e CONTROLLED CONTROLLE NOT (CCN). Estamos nos baseando nos estudos de Bennett [Ben73] sobre computadores reversíveis.

A porta $CN$ é um dispositivo com duas entradas $A$ e $B$ e duas saídas $A'$ e $B'$. Podemos resumir o funcionamento dessa porta da seguinte maneira: o valor de $A'$ é sempre igual ao valor de $A$. Já o valor de $B'$ depende dos valores de $A$ e de $B$. Se $A = 1$, então $B'$ vai ser a negação de $B$. Caso contrário, $B' = B$.

Analisando seu funcionamento, é fácil ver que esta porta é reversível. Além disso, podemos ver $B'$ como sendo a saída da função XOR aplicada sobre as entradas $A$ e $B$. A partir de uma porta $CN$, podemos construir também um circuito para a função FANOUT e um circuito para a função EXCHANGE.

Como as quatro operações que temos com as portas $N$ e $CN$ (NOT, XOR, FANOUT e EXCHANGE) não são suficientes para representar todas as funções, precisaremos de mais portas. Para que o conjunto fique completo, usaremos a porta $CCN$.

O funcionamento da porta $CCN$ é muito parecido com o da $CN$. Essa porta tem três entradas, $A$, $B$ e $C$, e três saídas, $A'$, $B'$ e $C'$. Analogamente à porta $CN$, teremos $A' = A$ e $B' = B$. Já o valor de $C'$ dependerá do valor de $A$ e $B$. Se $A = B = 1$, então $C'$ terá o valor inverso de $C$. Caso contrário, $C' = C$.

Esta porta é bastante poderosa. Se colocarmos sinal 1 na porta $A$, temos que $B$, $C$, $B'$ e $C'$ formam uma porta $CN$. Além disso, ao colocar sinal 0 em $C$, obtemos uma porta AND, tendo como entrada $A$ e $B$ e como saída $C'$.

Neste ponto, já temos as portas AND, NOT, FANOUT e EXCHANGE, que são suficientes para o que queremos. Logo, chegamos a um conjunto de portas que podem montar qualquer função de decisão desejada e que são todas reversíveis. Isso sugere que o poder computacional desses componentes não é inferior ao poder dos componentes usados para operações não-reversíveis.

# Referências Bibliográficas


[AGP94] W.R. Alford, A. Granville, and C. Pomerance.
There are infinitely many Carmichael numbers.
*Ann. of Math. (2)*, 139(3):703–722, 1994.

[AH92] L.M. Adleman and M.-D. Huang.
Primality testing and two dimensional abelian varieties over finite fields.
volume 1512 of *Lecture Notes in Mathematics*. Springer, Berlin, 1992.

[AKS02a] M. Agrawal, N. Kayal, and N. Saxena.
PRIMES is in P.
Preprint. `http://www.cse.iitk.ac.in/news/primality.pdf`, 2002.

[AKS02b] M. Agrawal, N. Kayal, and N. Saxena.
PRIMES is in P.
Revised. `http://www.cse.iitk.ac.in/news/primality_v3.pdf`, 2002.

[BBBV97] C.H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani.
Strengths and weaknesses of quantum computing.
*SIAM J. Comput.*, 26(5):1510–1523, 1997.

[BBC+95] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N.H. Margolus, P.W. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter.
Elementary gates for quantum computation.
*Physical Review A*, 52(5):3457–3467, 1995.

[Ben73] C.H. Bennett.
Logical reversibility of computation.
*IBM J. Res. Develop.*, 17:525–532, 1973.

[Ber98] D.J. Bernstein.
Detecting perfect powers in essentially linear time.
*Math. Comp.*, 67(223):1253–1283, 1998.

[BH97] G. Brassard and P. Høyer.
An exact quantum polynomial-time algorithm for Simon's problem.
In *Israel Symposium on Theory of Computing Systems*, pages 12–23, 1997.

[BLP93] J.P. Buhler, H.W. Lenstra, Jr., and C. Pomerance.
Factoring integers with the number field sieve.
In *The development of the number field sieve*, volume 1554 of *Lecture Notes in Math.*, pages 50–94. Springer, Berlin, 1993.

[Bre89] D.M. Bressoud.
*Factorization and Primality Testing.*
Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1989.

[BV93] E. Bernstein and U. Vazirani.
Quantum complexity theory.
In *Proceedings of the 25th ACM Symposium on the Theory of Computation*, pages 11–20, New York, 1993. ACM Press.

[BV97] E. Bernstein and U. Vazirani.
Quantum complexity theory.
*SIAM J. Comput.*, 26(5):1411–1473, 1997.

[Car12] R.D. Carmichael.
On composite numbers $p$ which satisfy the Fermat congruence $a^{p-1} \equiv 1 \pmod{p}$.
*Amer. Math. Monthly*, 19:22–27, 1912.

[CEMM98] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca.
Quantum algorithms revisited.
*R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci.*, 454(1969):339–354, 1998.

[Chu33] A. Church.
A set of postulates for the foundation of logic.
*Annals of Mathematics*, 25:839–864, 1933.

[Chu36] A. Church.
An unsolvable problem of elementary number theory.
*Annals of Mathematics*, 58:345–363, 1936.

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein.
*Introduction to Algorithms.*
MIT Press, Cambridge, MA, second edition, 2001.

[Cob65] A. Cobham.
The intrinsic computational difficult of functions.

In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science*, pages 24–30. North-Holland, 1965.

[Coo71] S. Cook.
The complexity of theorem proving procedures.
In *Proc. 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[Cop93] D. Coppersmith.
Modifications to the number field sieve.
*J. Cryptology*, 6(3):169–180, 1993.

[Cou00] S.C. Coutinho.
*Números inteiros e criptografia RSA*, volume 2 of *Série de Computação e Matemática.*
Instituto de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, second edition, 2000.

[Deu85] D. Deutsch.
Quantum theory, the Church-Turing principle and the universal quantum computer.
*Proc. Roy. Soc. London Ser. A*, 400(1818):97–117, 1985.

[Deu89] D. Deutsch.
Quantum computational networks.
*Proc. Roy. Soc. London Ser. A*, 425(1868):73–90, 1989.

[DH76] W. Diffie and M.E. Hellman.
New directions in cryptography.
*IEEE Trans. Information Theory*, IT-22(6):644–654, 1976.

[Edm65] J. Edmonds.
Paths, trees, and flowers.
*Canadian Journal of Mathematics*, 17:449–467, 1965.

[EJ96] A. Ekert and R. Jozsa.
Quantum computation and Shor's factoring algorithm.
*Rev. Mod. Phys.*, 68(3), 1996.

[Fey82] R. Feynman.
Simulating physics with computers.
*International Journal of Theoretical Physics*, 21(6 & 7):467–488, 1982.

[Fey96] R. Feynman.
*Feynman Lectures in Computation.*
Addison-Wesley Publishing Company, 1996.

[Fra89] J.B. Fraleigh.
*A First Course in Abstract Algebra.*
Addison-Wesley Publishing Company, fourth edition, 1989.

[GJ79] M.R. Garey and D.S. Johnson.
*Computers and Intractability: a Guide to the Theory of NP-Completeness.*
Freeman, 1979.

[GK86] S. Goldwasser and J. Kilian.
Almost all primes can be quickly certified.
In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 316–329. ACM Press, 1986.

[GKP94] R.L. Graham, D.E. Knuth, and O. Patashnik.
*Concrete Mathematics.*
Addison-Wesley Publishing Company, Reading, MA, second edition, 1994.
A foundation for computer science.

[Göd31] K. Gödel.
On formally undecidable propositions of Principia Mathematica and related systems.
*Monatshefte fur Math. und Physik*, 38:173–198, 1931.

[Hal42] P.R. Halmos.
*Finite Dimensional Vector Spaces.*
Annals of Mathematics Studies, no. 7. Princeton University Press, Princeton, N.J., 1942.

[HW54] G.H. Hardy and E.M. Wright.
*An Introduction to the Theory of Numbers.*
Oxford, at the Clarendon Press, 1954.
3rd ed.

[Joy] D.E. Joyce.
Mathematical problems by professor David Hilbert.
`http://aleph0.clarku.edu/~djoyce/hilbert/problems.html`.

[Kar72] R.M. Karp.
Reducibility among combinatorial problems.
In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.

[Kle36] S. Kleene.
General recursive functions of natural numbers.

*Mathematische Annalen*, 112:727–742, 1936.

[Kle52] S. Kleene.
*Introduction to Metamathematics.*
D. Van Nostrand, Princeton, NJ, 1952.

[Knu81] D.E. Knuth.
*The Art of Computer Programming. Vol. 2.*
Addison-Wesley Publishing Co., Reading, Mass., second edition, 1981.
Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing.

[KO62] A. Karatsuba and Y. Ofman.
Multiplication of multidigit numbers by automata (em russo).
*Dokl. Akad Nauk SSSR*, 145:293–294, 1962.

[KO63] A. Karatsuba and Y. Ofman.
Multiplication of multidigit numbers by automata.
*Soviet Physics-Doklady*, 7:595–596, 1963.

[Len87] H.W. Lenstra, Jr.
Factoring integers with elliptic curves.
*Ann. of Math. (2)*, 126(3):649–673, 1987.

[Lev73] L.A. Levin.
Universal sorting problems.
*Problems of Information Transmission*, 9:265–266, 1973.

[LLMP93] A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, and J.M. Pollard.
The number field sieve.
In *The development of the number field sieve*, volume 1554 of *Lecture Notes in Math.*, pages 11–42. Springer, Berlin, 1993.

[Lom02] S.J. Lomonaco, Jr.
A Rosetta Stone for quantum mechanics with an introduction to quantum computation.
In *Quantum computation: a grand mathematical challenge for the twenty-first century and the millennium (Washington, DC, 2000)*, volume 58 of *Proc. Sympos. Appl. Math.*, pages 3–65. Amer. Math. Soc., Providence, RI, 2002.

[Mil75] G.L. Miller.
Riemann's hypothesis and tests for primality.
In *Seventh Annual ACM Symposium on Theory of Computing (Albuquerque, N.M., 1975)*, pages 234–239. Assoc. Comput. Mach., New York, 1975.

[Mil76] G.L. Miller.
Riemann's hypothesis and tests for primality.
*J. Comput. System Sci.*, 13(3):300–317, 1976.
Working papers presented at the ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N.M., 1975).

[MK01] C.G. Moreira and Y. Kohayakawa.
*Tópicos em combinatória contemporânea.*
Publicações Matemáticas do IMPA. Instituto de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, 2001.
23º Colóquio Brasileiro de Matemática.

[MM04] D.C. Marinescu and G.M. Marinescu.
Lectures on quantum computing.
`http://www.cs.ucf.edu/~dcm/Fall2003Class-QC/QCTextBookIndex.htm`, 2004.

[Mon80] L. Monier.
Evaluation and comparison of two efficient probabilistic primality testing algorithms.
*Theoret. Comput. Sci.*, 12(1):97–108, 1980.

[MR95] R. Motwani and P. Raghavan.
*Randomized Algorithms.*
Cambridge University Press, Cambridge, 1995.

[Pap94] C.H. Papadimitriou.
*Computational Complexity.*
Addison-Wesley Publishing Company, Reading, MA, 1994.

[Pol93] J.M. Pollard.
Factoring with cubic integers.
In *The development of the number field sieve*, volume 1554 of *Lecture Notes in Math.*, pages 4–10. Springer, Berlin, 1993.

[Pom96] C. Pomerance.
A tale of two sieves.
*Notices Amer. Math. Soc.*, 43(12):1473–1485, 1996.

[Pos36] E. Post.
Finite combinatory process.
*Journal of Symbolic Logic*, 1:103–105, 1936.

[Pra75] V.R. Pratt.
Every prime has a succinct certificate.
*SIAM J. Comput.*, 4(3):214–220, 1975.

[Pre04] J. Preskill.

Lecture notes for Physics 219/Computer Science 219.
`http://www.theory.caltech.edu/people/preskill/ph229`, 2004.

[Pru81] E. Prugovečki.
*Quantum Mechanics in Hilbert Space*, volume 92 of *Pure and Applied Mathematics*.
Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, second edition, 1981.

[Rab80] M.O. Rabin.
Probabilistic algorithm for testing primality.
*J. Number Theory*, 12(1):128–138, 1980.

[Ros00] K.H. Rosen.
*Elementary Number Theory and its Applications*.
Addison-Wesley Publishing Company, Reading, MA, fourth edition, 2000.

[RP00] E. Rieffel and W. Polak.
Introduction to quantum computing.
*ACM Computing Surveys*, 32(3):300–335, 2000.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman.
A method for obtaining digital signatures and public-key cryptosystems.
*Comm. ACM*, 21(2):120–126, 1978.

[Sav70] W.J. Savitch.
Relationships between nondeterministic and deterministic tape complexities.
*Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Sch82] A. Schönhage.
Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients.
In *Computer algebra (Marseille, 1982)*, volume 144 of *Lecture Notes in Comput. Sci.*, pages 3–15. Springer, Berlin, 1982.

[Sho94] P.W. Shor.
Algorithms for quantum computation: discrete logarithms and factoring.
In *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pages 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994.

[Sho97] P.W. Shor.

Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.
*SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Sim97] D.R. Simon.
On the power of quantum computation.
*SIAM J. Comput.*, 26(5):1474–1483, 1997.

[SS71] A. Schönhage and V. Strassen.
Schnelle Multiplikation grosser Zahlen.
*Computing (Arch. Elektron. Rechnen)*, 7:281–292, 1971.

[TI97] L.N. Trefethen and D. Bau III.
*Numerical Linear Algebra.*
Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[Tur36] A. Turing.
On computable numbers with an application to the entscheidungsproblem.
*Proc. London Math. Soc.*, 2:230–265, 1936.

[Tur37] A. Turing.
Rectifications to 'on computable numbers…'.
In *Proc. London Mathematical Society*, volume 4, pages 544–546, 1937.

# Índice Remissivo