Quantum Computing

Summary

Fabian Damken

July 21, 2022

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Contents

List of Figures

List of Tables

List of Algorithms

1 Preliminaries

In this chapter we discuss the groundwork for the upcoming topics. Along with these subjects, basic knowledge from linear algebra is required.

1.1 Complex Numbers

One of the underlying principles of quantum mechanics (QM) and therefore quantum computing (QC), too, are complex numbers. This section summarizes some results for them *very briefly*.

Let $z = a + ib \in \mathbb{C}$ be a complex number with the real and imaginary components $\mathrm{Re}(z) = a, \mathrm{Im}(z) = b \in \mathbb{R}$. Its magnitude is

$$|z| \coloneqq \sqrt{a^2 + b^2} = \sqrt{zz^*}$$

with the *complex conjugate* $z^* = a - ib$. The complex conjugate is distributive over addition and multiplication[1], i.e., $(z_1 + z_2)^* = z_1^* + z_2^*$ and $(z_1 z_2)^* = z_1^* z_2^*$ holds for two complex numbers $z_1, z_2 \in \mathbb{C}$. Any complex number can also be written in polar form $z = re^{i\varphi}$ with magnitude

$$|z| = \sqrt{zz^*} = \sqrt{re^{i\varphi} re^{-i\varphi}} = \sqrt{r^2 e^{i\varphi - i\varphi}} = \sqrt{r^2} = |r|.$$

Definition 1 (n-th Root of Unity)**.** We call the special complex number $\omega_n = e^{2\pi i/n}$ the *n-th root of unity.*

Theorem 1. *Let ω_n be the n-th root of unity with $n > 1$. Then $\sum_{k=0}^{n-1} \omega_n^{ak} = 0$ holds for every constant $a \in \mathbb{Z}$.*

Proof.

$$\sum_{k=0}^{n-1} \omega_n^{ak} = \sum_{k=0}^{n-1} (\omega_n^a)^k = \frac{1 - (\omega_n^a)^n}{1 - \omega_n^a} = \frac{1 - e^{2ia\pi}}{1 - \omega_n^a} = \frac{1 - 1}{1 - \omega_n^a} = \frac{0}{1 - \omega_n^a} = 0$$

□

1.2 Continued Fraction Expansion

Let $x \in (0, 1)$ be a real number[2]. Then we can express this number as its *continued fraction expansion (CFE)*

$$x = \cfrac{1}{a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots}}}$$

where $a_0, a_1, \cdots \in \mathbb{N}^+$. The CFE of x is finite iff x is rational. The sums

$$\frac{1}{a_0} \qquad \frac{1}{a_0 + \frac{1}{a_1}} \qquad \frac{1}{a_0 + \cfrac{1}{a_1 + \frac{1}{a_2}}} \qquad \cdots$$

[1] For other useful properties, see `https://en.wikipedia.org/wiki/Complex_conjugate#Properties`.

[2] Note that the restriction on the interval $(0, 1)$ is purely for convenience as we only have x's between zero and one down the line. It is also possible to extend continued fraction expansions to $\mathbb{R}$.

are called *partial sums.* For calculating $a_0, a_1, \ldots$, let

$$x_0 \coloneqq \frac{1}{a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \cdots}}} \qquad x_1 \coloneqq \frac{1}{a_1 + \frac{1}{a_2 + \cdots}} \qquad x_2 \coloneqq \frac{1}{a_2 + \cdots} \qquad \cdots$$

then the coefficients are $a_i = [1/x_i]$, where the brackets indicate the integral part, i.e., the part in front of the decimal. If for any j, $x_j = 0$, the CFE terminates and the number is exactly represented.

Example 1. *Let* $x = 11\,490/2^{14} \approx 0.701294$*. Then the CFE is calculated as follows:*

i	x_i	$1/x_i$	a_i
0	0.701 294	1.425 94	1
1	0.425 94	2.347 77	2
2	0.347 77	2.875 44	2
3	0.875 44	1.142 28	1
4	0.142 28	7.028 30	7
5	0.028 30	35.3333	35
6	0.333 33	3	3
7	0		

The final CFE is therefore

$$x = \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{7 + \cfrac{1}{35 + \frac{1}{3}}}}}}}$$

with the coefficients $(a_0, a_1, a_2, a_3, a_4, a_5, a_6) = (1, 2, 2, 1, 7, 35, 3)$.

2 Postulates of Quantum Mechanics

In this chapter we discuss the postulates of QM and some important protocols and results in QC such as the *no-cloning theorem.* This theorem states that it is impossible to copy a quantum state!

2.1 States

In classical computing, a bit is either 0 or 1. A quantum bit, a *qubit,* however, is more general and has the basis states $|0\rangle$ and $|1\rangle$. The states are formed by basis vectors $|0\rangle = (1,0)^\dagger$ and $|1\rangle = (0,1)^\dagger$. More generally, an arbitrary quantum state $|\psi\rangle$ can be a combination of the basis states, $|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle$, a *superposition* (with complex coefficients $c_0, c_1 \in \mathbb{C}$). However, the state has to be normalized, i.e., $|\langle\psi|\psi\rangle|^2 = 1$. The left part of this inner product is called a *bra* vector representing the conjugate transpose of the right side, the *ket* vector.

The following postulate digests this idea more formally.

Postulate 1 (Quantum State)**.** *Any closed physical system can be associated with a Hilbert space $\mathcal{H}$. The state of the system is completely described by a state vector $|\psi\rangle = \sum_{i=0}^{d-1} c_i |i\rangle$ with $\sum_{i=0}^{d-1} |c_i|^2 = 1$ where $\{|i\rangle\}_{i=0}^{d-1}$ forms a basis of $\mathcal{H}^d$.*

Remark 1. *The basis is not confined to the* computational basis $\{|0\rangle, |1\rangle\}$*, although this basis is often used. It may be any other orthonormal basis of $\mathcal{H}$, see section 2.2.1. For basis of Hilbert spaces with $d > 2$, see section 2.4.*

Instead of writing out the complex coefficients c_0 and c_1, we can also parameterize an arbitrary superposition with angles $\gamma, \varphi, \theta \in \mathbb{R}$:

$$|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right).$$

However, as we will see in section 2.3, a global phase such as $e^{i\gamma}$ vanishes in all important calculations as $e^{i\gamma}e^{-i\gamma} = 1$. Hence, we can also parameterize any state with just two angles $\varphi \in (0, 2\pi]$ and $\theta \in (0, \pi]$:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle.$$

Checking that this state is actually normalized is straightforward:

$$\begin{aligned}
\langle\psi|\psi\rangle &= \left(\cos\frac{\theta}{2}\langle 0| + e^{-i\varphi}\sin\frac{\theta}{2}\langle 1|\right)\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \\
&= \cos\frac{\theta}{2}\langle 0| + e^{-i\varphi}\sin\frac{\theta}{2}\langle 1|\right)\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \\
&= \cos^2\frac{\theta}{2}\underbrace{\langle 0|0\rangle}_{=1} + e^{i\varphi}\cos\frac{\theta}{2}\sin\frac{\theta}{2}\underbrace{\langle 0|1\rangle}_{=0} + e^{-i\varphi}\cos\frac{\theta}{2}\sin\frac{\theta}{2}\underbrace{\langle 1|0\rangle}_{=0} + e^{i\varphi}e^{-i\varphi}\sin\frac{\theta}{2}\sin\frac{\theta}{2}\underbrace{\langle 1|1\rangle}_{=1} \\
&= \cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2} = 1.
\end{aligned}$$

Note that in this case $\langle\psi|\psi\rangle = 1$, so $|\langle\psi|\psi\rangle|^2 = 1$ holds, too, and we can drop the absolute-square. In most of the following discussions where we need explicit parametrization, we confine ourselves to real coefficients, i.e., $\varphi = 0$. This simplifies the discussion as now there is only one parameter θ.

2.2 Evolution

The evolution of quantum states, i.e., how they pass between states, is described by linear transformations U, also called *gates*. These gates transform a quantum state $|\psi\rangle$ into another quantum state $|\psi'\rangle$. In quantum circuits, we denote an application of U_1 and then U_2 to a state $|\psi\rangle$, i.e., $U_2U_1|\psi\rangle$, as:

$$|\psi\rangle \;\text{—}\boxed{U_1}\text{—}\boxed{U_2}\text{—}\; U_2U_1|\psi\rangle$$

Postulate 2 (State Evolution)**.** *The evolution $|\psi(t_0)\rangle \xrightarrow{U} |\psi(t)\rangle$ of a closed physical system is described by a unitary transformation $UU^\dagger = \mathbb{1}$.*

Theorem 2 (Unitarity of Quantum Gates)**.** *A linear quantum gate U is unitary, i.e., $UU^\dagger = \mathbb{1}$.*

Proof. □

2.2.1 Gates

In this section we collect the most important single-qubit gates. They are summarized in Table 2.1 and their semantics are given in the caption.

Theorem 3 (Decomposition of Two-By-Two Unitary Matrices)**.** *Every unitary matrix $U \in \mathbb{C}^{2\times 2}$ can be decomposed into three rotations as $U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta)$.*

From this theorem, one might think that it is necessary to implement every rotation in the lab for a universal quantum computer. Fortunately, this is not the case! As we will see in chapter 4, only three gates are necessary to implement arbitrary rotations.

The Hadamard Gate

A special gate we regularly use is the Hadamard gate H. As shown in Table 2.1 transforms it the computational basis states into an equal superposition of themselves:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) =: |+\rangle \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =: |-\rangle \tag{2.1}$$

As these states are extremely important, we often denote them as $|+\rangle$ and $|-\rangle$ which are again basis states of the Hilbert space as the following hold:

$$\langle +|+\rangle = 1 \qquad \langle +|-\rangle = 0 \qquad \langle -|+\rangle = 0 \qquad \langle -|-\rangle = 1$$

As these vectors are the Eigenvectors of X, we often call this basis the X-basis while the computational basis is also often called Z-basis.

Of course, we might also want to apply the Hadamard gate to a composite state (see section 2.4 for an introduction to composite states). As the application and transformation into the computational basis "by hand" is quite tedious,

$$H^{\otimes 2}|00\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle),$$

we often make use of the following result.

U	$U\|0\rangle$	$U\|1\rangle$	$U\|+\rangle$	$U\|-\rangle$
$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = HZH$	$\|1\rangle$	$\|0\rangle$	$\|+\rangle$	$-\|-\rangle$
$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \equiv XZ$	$i\|1\rangle$	$-i\|0\rangle$	$-i\|-\rangle$	$i\|+\rangle$
$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\|0\rangle$	$-\|1\rangle$	$\|-\rangle$	$\|+\rangle$
$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	$\|+\rangle$	$\|-\rangle$	$\|0\rangle$	$\|1\rangle$
$R_y(\gamma) = \begin{bmatrix} c_\gamma & -s_\gamma \\ s_\gamma & c_\gamma \end{bmatrix}$	$c_\gamma\|0\rangle + s_\gamma\|1\rangle$	$-s_\gamma\|0\rangle + c_\gamma\|1\rangle$	$c_\gamma\|+\rangle - s_\gamma\|-\rangle$	$s_\gamma\|+\rangle + c_\gamma\|-\rangle$
$R_z(\beta) = \begin{bmatrix} e^{i\beta/2} & 0 \\ 0 & e^{-i\beta/2} \end{bmatrix}$	$e^{i\beta/2}\|0\rangle$	$e^{-i\beta/2}\|1\rangle$	$e^{i\beta/2}\|0\rangle + e^{-i\beta/2}\|1\rangle$	$e^{i\beta/2}\|0\rangle - e^{-i\beta/2}\|1\rangle$

Table 2.1: Common qubit gates and their effect on the computational and Hadamard basis. For brevity, let $c_\gamma := \cos(\gamma/2)$ and $s_\gamma := \sin(\gamma/2)$. The gates have the following effects in the computational basis: X implements a logical not, Y combines a phase-flip and logical not, Z implements a phase-flip, H creates an equal superposition, $R_y(\gamma)$ rotates around an arbitrary angle γ, and $R_z(\beta)$ adds a phase. In Hadamard basis, the gates have the following effects: X implement a phase-flip, Y combined a phase-flip and logical not, Z implements a logical not, H creates an equal superposition, $R_y(\gamma)$ rotates around an arbitrary angle γ, and $R_z(\beta)$ adds a phase.

Theorem 4 (Hadamard Gate on Basis States)**.** *Let $|x\rangle$ be any basis state with n qubits identified by the bit string $x \in \{0,1\}^n$. Then applying Hadamard gates on the individual qubits yields*

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

where z sums over all possible n-bit strings, i.e., the basis states, and $x \cdot y$ is the bit-wise dot-product of x and y modulo 2.

Proof. We proof this by induction. For $n = 1$, we have

$$H|x\rangle = \frac{1}{\sqrt{2}}\big((-1)^{x\cdot 0}|0\rangle + (-1)^{x\cdot 1}|1\rangle\big) = \frac{1}{\sqrt{2}}\big(|0\rangle + (-1)^x|1\rangle\big)$$

matching (2.1). Assume now that the statement holds for some n and let $x =: x_1 x_:$ be a splitting of the $(n+1)$-bit string such that $x_1 \in \{0,1\}$ and $x_: \in \{0,1\}^n$ and analogous for $y =: y_1 y_:$. We show that the

statement also holds for $n+1$:

$$\begin{aligned}
H^{\otimes(n+1)}\left|x\right\rangle &= H^{\otimes(n+1)}\left|x_1\right\rangle\left|x_{:}\right\rangle = H\left|x_1\right\rangle \otimes H^{\otimes n}\left|x_{:}\right\rangle \\
&\overset{(\dagger)}{=} \frac{1}{\sqrt{2}}\big(\left|0\right\rangle + (-1)^{x_1}\left|1\right\rangle\big) \otimes \frac{1}{\sqrt{2^n}} \sum_{y_{:}\in\{0,1\}^n} (-1)^{x_{:}\cdot y_{:}}\left|y_{:}\right\rangle \\
&= \frac{1}{\sqrt{2}}\big((-1)^{x_1\cdot 0}\left|0\right\rangle + (-1)^{x_1\cdot 1}\left|1\right\rangle\big) \otimes \frac{1}{\sqrt{2^n}} \sum_{y_{:}\in\{0,1\}^n} (-1)^{x_{:}\cdot y_{:}}\left|y_{:}\right\rangle \\
&= \frac{1}{\sqrt{2}} \sum_{y_1\in\{0,1\}} (-1)^{x_1\cdot y_1}\left|y_1\right\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{y_{:}\in\{0,1\}^n} (-1)^{x_{:}\cdot y_{:}}\left|y_{:}\right\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{\substack{y_1\in\{0,1\}\\ y_{:}\in\{0,1\}^n}} (-1)^{x_1\cdot y_1}\left|y_1\right\rangle \otimes (-1)^{x_{:}\cdot y_{:}}\left|y_{:}\right\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{\substack{y_1\in\{0,1\}\\ y_{:}\in\{0,1\}^n}} (-1)^{x_1\cdot y_1 \oplus x_{:}\cdot y_{:}}\left|y_1\right\rangle\left|y_{:}\right\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{y\in\{0,1\}^{n+1}} (-1)^{x\cdot y}\left|y\right\rangle
\end{aligned}$$

We used the induction hypothesis in step $(\dagger)$. This completes the proof and shows that the formulation of the Hadamard gates holds for all n. □

2.3 Measurement

We will now discuss the last postulate of QM which is concerned with *measurements.* The central result is that measuring a quantum system is inherently *probabilistic,* i.e., the outcome of a measurement is not deterministic and truly random. For any quantum state $\left|\psi\right\rangle$, the probability of measuring an outcome v_i is given by the absolute-square of the inner product between the "measurement state" $\left|v_i\right\rangle$ and the state $\left|\psi\right\rangle$:

$$P(i) = |\langle v_i|\psi\rangle|^2.$$

The value of this inner product (without the absolute-square) is called the *probability amplitude* and can be negative or even complex. Immediately after a measurement, the state $\left|\psi\right\rangle$ collapses into a post-measurement state $\left|\psi'\right\rangle$. This post-measurement state is

$$\left|\psi'\right\rangle = \frac{M_i\left|\psi\right\rangle}{N_i}$$

where $M_i = \left|v_i\right\rangle\!\left\langle v_i\right|$ and $N_i = \sqrt{P(i)}$ are the *measurement operator* and *normalization constant,* respectively. These results are digested in the following postulate.

Postulate 3 (Quantum Measurement)**.** *Quantum measurements are described by a collection of measurement operators $\{M_i\}$ where i indicated the outcome of the experiment. Let $\left|\psi\right\rangle$ be the state before the measurement, then the state immediately after the measurement is $\left|\psi'\right\rangle = M_i\left|\psi\right\rangle / N_i$ where $N = \sqrt{P(i)}$ is for normalization.*

Theorem 5 (Measurement of Pure Quantum States)**.** *For* pure *states $\left|\psi\right\rangle$, the post-measurement state $\left|\psi'\right\rangle$ after a measurement of $\left|v_i\right\rangle$ is $\left|\psi'\right\rangle = \left|v_i\right\rangle$.*

Proof.

$$\frac{M_i\left|\psi\right\rangle}{N_i} = \frac{\left|v_i\right\rangle\!\left\langle v_i\right|\left|\psi\right\rangle}{\sqrt{P(i)}} = \frac{\left|v_i\right\rangle\!\left\langle v_i\right|\left|\psi\right\rangle}{\sqrt{|\langle v_i|\psi\rangle|^2}} = \frac{\left|v_i\right\rangle\left\langle v_i|\psi\right\rangle}{|\langle v_i|\psi\rangle|} = \left|v_i\right\rangle \frac{\langle v_i|\psi\rangle}{|\langle v_i|\psi\rangle|} \equiv \left|v_i\right\rangle$$

□

2.4 Composite Systems and Tensor Products

As in classical computing where we are concerned with more than one bit, QC also works with more than one qubit. The formalism for this are *tensor products* $\mathcal{H}^2 \otimes \mathcal{H}^2$ between the Hilbert spaces of the individual qubits. Its basis vectors are also constructed using tensor products:

$$|0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \qquad |0\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \qquad |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \qquad |1\rangle \otimes |1\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

For two single-qubit operators $A = \begin{bmatrix}a_{00} & a_{01}\\ a_{10} & a_{11}\end{bmatrix}$ and $B = \begin{bmatrix}b_{00} & b_{01}\\ b_{10} & b_{11}\end{bmatrix}$, the tensor product is carried out as

$$A \otimes B = \begin{bmatrix}a_{00} & a_{01}\\ a_{10} & a_{11}\end{bmatrix} \otimes \begin{bmatrix}b_{00} & b_{01}\\ b_{10} & b_{11}\end{bmatrix} = \begin{bmatrix} a_{00}\begin{bmatrix}b_{00} & b_{01}\\ b_{10} & b_{11}\end{bmatrix} & a_{01}\begin{bmatrix}b_{00} & b_{01}\\ b_{10} & b_{11}\end{bmatrix} \\ a_{10}\begin{bmatrix}b_{00} & b_{01}\\ b_{10} & b_{11}\end{bmatrix} & a_{11}\begin{bmatrix}b_{00} & b_{01}\\ b_{10} & b_{11}\end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{01}b_{00} & a_{01}b_{01} \\ a_{00}b_{10} & a_{00}b_{11} & a_{01}b_{10} & a_{01}b_{11} \\ a_{10}b_{00} & a_{10}b_{01} & a_{11}b_{00} & a_{11}b_{01} \\ a_{10}b_{10} & a_{10}b_{11} & a_{11}b_{10} & a_{11}b_{11} \end{bmatrix}$$

with some abuse of notation. This definition has the effect of applying unitary A to the first and unitary B to the second qubit in a tensor-multiplied Hilbert space, i.e.,

$$(A \otimes B)(|\psi\rangle_1 \otimes |\psi\rangle_2) = (A\,|\psi\rangle_1) \otimes (B\,|\psi\rangle_2)$$

For brevity, we often write product state as $|0\rangle \otimes |1\rangle \doteq |0\rangle\,|1\rangle \doteq |01\rangle$ and the application of product operators as $(A \otimes B)(|0\rangle \otimes |1\rangle) \doteq A \otimes B\,|0\rangle \otimes |1\rangle = A_1 B_2\,|01\rangle$. As long as it is clear which unitary is applied to which qubit, a variety of notations may be used. For brevity, we also often write $|\psi\rangle^{\otimes N} \doteq \underbrace{|\psi\rangle \otimes \cdots \otimes |\psi\rangle}_{N \text{ times}}$ and the same for gates.

2.4.1 Entanglement

A *composite* or *product* state is a state $|\psi_{12}\rangle$ that can be written as the product of two individual states $|\psi_1\rangle = \alpha_1\,|0\rangle + \beta_1\,|1\rangle$ and $|\psi_2\rangle = \alpha_2\,|0\rangle + \beta_2\,|1\rangle$:

$$|\psi_{12}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = (\alpha_1\,|0\rangle + \beta_1\,|1\rangle) \otimes (\alpha_2\,|0\rangle + \beta_2\,|1\rangle) = \alpha_1\alpha_2\,|00\rangle + \alpha_1\beta_2\,|01\rangle + \beta_1\alpha_2\,|10\rangle + \beta_1\beta_2\,|11\rangle$$

However, there are states that cannot be written like this!

Definition 2 (Entangled State)**.** A quantum state $|\psi_{12}\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ is called *entangled* if there are no states $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$ such that $|\psi_{12}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$.

Theorem 6 (Simple Entangled States)**.** *All states* $|\psi_\theta\rangle = \cos\frac{\theta}{2}\,|00\rangle + \sin\frac{\theta}{2}\,|11\rangle$*,* $\theta \in (0, \pi/2]$ *are entangled.*

Proof. Let $|\psi_1\rangle \coloneqq \alpha_1\,|0\rangle + \beta_1\,|1\rangle$ and $|\psi_2\rangle \coloneqq \alpha_2\,|0\rangle + \beta_2\,|1\rangle$ with coefficients $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{C}$. Assume that $|\psi_\theta\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. Hence,

$$|\psi_\theta\rangle = \alpha_1\alpha_2\,|00\rangle + \alpha_1\beta_2\,|01\rangle + \beta_1\alpha_2\,|10\rangle + \beta_1\beta_2\,|11\rangle \overset{!}{=} \cos\frac{\theta}{2}\,|00\rangle + \sin\frac{\theta}{2}\,|11\rangle$$

By comparing coefficients, all of the following must hold: $\alpha_1\alpha_2 =\neq 0$, $\beta_1\beta_2 =\neq 0$, and $\alpha_1\beta_2 = \beta_1\alpha_2 = 0$. From the first two constraints it follows that all coefficients must be non-zero which contradicts the last constraint. Hence, the state is entangled. □

One important special case of this result is the *Bell state* $|\Phi^+\rangle \coloneqq \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ which we will study further in chapter 7.

Multipartite

So far, we only studied entanglement of two parties $\mathcal{H}_1$ and $\mathcal{H}_2$. However, it is also possible to describe entanglement between three or more parties. For three parties $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$, there can be a variety of different entanglements:

$$|\psi_{123}\rangle = |\psi_{12}\rangle \otimes |\psi_3\rangle \qquad |\psi_{123}\rangle = |\psi_1\rangle \otimes |\psi_{23}\rangle \qquad |\psi_{123}\rangle = |\psi_2\rangle \otimes |\psi_{13}\rangle$$

For more than two parties, a state $|\psi\rangle_{123}$ that cannot be expressed as a product of its components is called *genuine multipartite entangled (GME)*. To check whether some state is GME can be done explicitly analogous to the above proof of two-party entanglement by checking all the above cases along with

$$|\psi 123\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle \,.$$

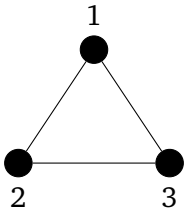
However, for N qubits the (potentially) entangled state has 2^N coefficients! The complexity of this checking is therefore $\mathcal{O}(\text{scary})$. There are, however, less straightforward, but easier-to-check procedures for validating whether a state is GME, but these are out of scope of this course.

Graph States

Although general methods for checking GME is out of scope, we will still look at the most famous example: *graph states*. Graph states a multi-qubit states corresponding to the mathematical structure of a graph. Let $G = (V, E)$ be a graph with vertices V and edges E. Then the corresponding multi-qubit state is

$$|G\rangle = \prod_{e \in E} CZ_e \, |+\rangle^{\otimes |V|}$$

where $CZ_e = \mathrm{diag}(1, 1, 1, -1)$ is a controlled-Z-gate (see subsection 2.4.2) acting on the qubits of the edge. These graph states allowed for a new language to reason about quantum states. For instance, when measuring the first qubit of the following graph state in Z-basis,



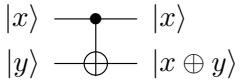
it just disappears, dropping the connections to the second and third qubit:

Similar rules exist for other measurements, but these are again out of scope for this course.

2.4.2 Multi-Qubit Gates

So far, we only discussed local gates acting on a single qubit (remember, gates combined with tensor products are applied on each gate individually). While this already allows some calculations, it does not allow interplay of multiple qubits or generation of entangled states which are very important for various protocols (see section 2.5). Hence, we need *multi-qubit gates* U that cannot be written as the product of local gates, i.e., $U \neq U_1 \otimes \cdots \otimes U_N$.

CNOT-Gate The simplest is the CNOT-gate:

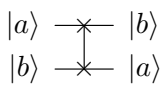


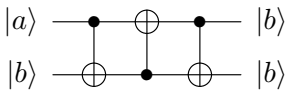
$$CNOT_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Input	Output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

This gate is a *controlled* gate and applied the X-gate to the second qubit iff the first qubit is 1. The indices $CNOT_{ij}$ indicate that the gate is acting on the j-th qubit (the *target*) and controlled by the i-th qubit. This gate can be extended to more than two qubits (with $n-1$ control qubits and a single target). For $n=3$, it is called the *Toffoli gate* which can be used to represent classical logical operations like logical not, and, or, and not-and.

SWAP-Gate Another important two-qubit gate is the SWAP-gate



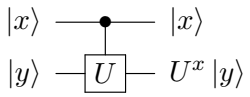


which simply switches the state of two qubits. The circuit on the right is the implementation of the SWAP-gate. Showing their equivalence is straightforward:

$$|00\rangle \xrightarrow{CNOT_{12}} |00\rangle \xrightarrow{CNOT_{21}} |00\rangle \xrightarrow{CNOT_{12}} |00\rangle \qquad |01\rangle \xrightarrow{CNOT_{12}} |01\rangle \xrightarrow{CNOT_{21}} |11\rangle \xrightarrow{CNOT_{12}} |10\rangle$$
$$|10\rangle \xrightarrow{CNOT_{12}} |11\rangle \xrightarrow{CNOT_{21}} |01\rangle \xrightarrow{CNOT_{12}} |01\rangle \qquad |11\rangle \xrightarrow{CNOT_{12}} |10\rangle \xrightarrow{CNOT_{21}} |10\rangle \xrightarrow{CNOT_{12}} |11\rangle$$

As unitary transformations are linear, we almost always only have to show the equivalence for the basis states as every state can be expressed as a superposition of them. This simplifies a lot of derivations! As the above circuit implements swapping for the basis states, it is a valid implementation of the SWAP-gate.

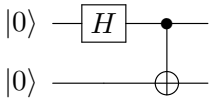
Controlled-U-Gate Note that any gate U can be used in a controlled fashion:



$$CU_{12} = \begin{bmatrix} \mathbb{1} & 0 \\ 0 & U \end{bmatrix}$$

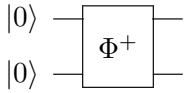
To implement this gate in practice, it can be decomposed into CNOT-gates and single-qubit gates (see **??**).

Preparing the Bell State Equipped with these tools, we can prepare the Bell state $|\Phi^+\rangle$ with the following circuit:



$$|00\rangle \xrightarrow{H_1} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|0\rangle \xrightarrow{CNOT_{12}} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \left|\Phi^+\right\rangle$$

For brevity, we will write



from now on whenever a Bell state is prepared between two qubits. Also, we will leave out the explicit derivation of the Bell state will from derivations.

2.5 Protocols

In this section we discuss some essential protocols in QC and the no-cloning theorem. These protocols are not complete algorithms (which are discussed in chapter 5), but illustrate essential ideas supporting some of the algorithms.

2.5.1 No-Cloning

While the no-cloning theorem is not really a protocol, it is an extremely important result for QC and thus also covered here.

Theorem 7 (No-Cloning)**.** *Let $|\psi\rangle$ be some state. Then there exists no U such that $U|\psi\rangle|0\rangle = |\psi\rangle|\psi\rangle$. That is, no circuit exists that copies an arbitrary quantum state.*

Proof. Assume that U is a cloning circuit and let $|\psi\rangle$ and $|\phi\rangle$ be arbitrary states. Then we can compute

$$(\langle\phi|\langle\phi|)(|\psi\rangle|\psi\rangle) = \langle\phi|\psi\rangle\langle\psi|\phi\rangle = (\langle\phi|\psi\rangle)^2.$$

However, we can also express the composite states as $|\phi\rangle|\phi\rangle = U|\phi\rangle|0\rangle$ and $|\psi\rangle|\psi\rangle = U|\psi\rangle|0\rangle$ using the definition of the cloning circuit U. Hence,

$$\langle 0|\langle\phi|\underbrace{U^\dagger U}_{=\mathbb{1}}|\psi\rangle|0\rangle = \langle 0|\langle\phi|\psi\rangle|0\rangle = \langle 0|0\rangle\langle\phi|\psi\rangle = \langle\phi|\psi\rangle.$$

Therefore, $(\langle\phi|\psi\rangle)^2 = \langle\phi|\psi\rangle$ holds. The states $|\phi\rangle$ and $|\psi\rangle$ are therefore orthogonal, $\langle\phi|\psi\rangle = 0$, or equal, $\langle\phi|\psi\rangle = 1$. This corresponds to classical data (either 0 or 1) and no arbitrary quantum states. Hence, there exists no such U. □

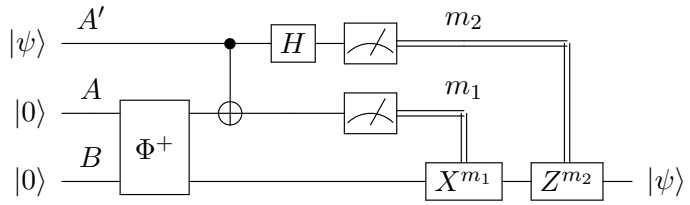
This theorem is a fundamental result of QC and hinders some algorithms down the line. But it is not new! In fact, the no-cloning theorem is *equivalent* to Heisenberg's uncertainty principles stating that for any quantum system there exist two properties which cannot both be measured with certainty. Proofing this equivalence would go as follows (proofing both directions using contraposition):

- From no-cloning to Heisenberg: if Heisenberg's uncertainty principle would be false, we could measure everything with certainty and thus prepare a second state simply by transferring the measured data, violating the no-cloning theorem.
- From Heisenberg to no-cloning: if the no-cloning theorem would be false, we could copy an arbitrary quantum state an arbitrary number of times and thus measure the state with arbitrary precision, violating Heisenberg's uncertainty principle.

2.5.2 Quantum Teleportation

With *quantum teleportation,* it is possible to teleport an arbitrary quantum state from one position to another (e.g., from Alice's to Bob's lab) using entanglement. Both parties (Alice and Bob) previously shared a Bell state $|\Phi^+\rangle$ and now Alice wants to transmit her state $|\psi\rangle$ over to Bob, but they cannot meet and have no secure communication channel. However, Alice can publicly announce two classical bits of information that Bob will read.

Consider the following circuit:



(2.2)

Note how the state $|\psi\rangle$ is teleported from qubit A to qubit B. Also note that the state is not cloned as Alice's measurement destroys her copy. To see that the above circuit actually copies the state, we can simply calculate what it does to the circuit. Let $|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle$ be the qubit to be copied. Right before the measurements, the system has the following state:

$$
\begin{aligned}
& (c_0 |0\rangle + c_1 |1\rangle)_{A'} |00\rangle_{AB} \\
\xrightarrow{\Phi^+_{AB}} \quad & \frac{1}{\sqrt{2}}(c_0 |0\rangle + c_1 |1\rangle)_{A'}(|00\rangle + |11\rangle)_{AB} \\
\xrightarrow{CNOT_{A'A}} \quad & \frac{1}{\sqrt{2}}\big(c_0 |0\rangle_{A'} (|00\rangle + |11\rangle)_{AB} + c_1 |1\rangle_{A'} (|10\rangle + |01\rangle)_{AB}\big) \\
\xrightarrow{H_{A'}} \quad & \frac{1}{\sqrt{2}}\big(c_0 |+\rangle_{A'} (|00\rangle + |11\rangle)_{AB} + c_1 |-\rangle_{A'} (|10\rangle + |01\rangle)_{AB}\big) \\
= \quad & \frac{1}{2}\big(c_0(|0\rangle + |1\rangle)_{A'}(|00\rangle + |11\rangle)_{AB} + c_1(|0\rangle - |1\rangle)_{A'}(|10\rangle + |01\rangle)_{AB}\big) \\
= \quad & \frac{1}{2}\big(|00\rangle_{A'A} (c_0 |0\rangle + c_1 |1\rangle)_B + |01\rangle_{A'A} (c_0 |1\rangle + c_1 |0\rangle)_B \\
& \qquad + |10\rangle_{A'A} (c_0 |0\rangle - c_1 |1\rangle) + |11\rangle_{A'A} (c_0 |1\rangle - c_1 |0\rangle)_B\big)
\end{aligned}
$$

When now measuring the first two qubits, the following outcomes and post-measurement states are present, and the corresponding corrections have to be applied to recover $|\psi\rangle$:

m_1	m_2	$\|\psi'\rangle_B$	Correction
0	0	$c_0 \|0\rangle + c_1 \|1\rangle$	$\mathbb{1}$
0	1	$c_0 \|1\rangle + c_1 \|0\rangle$	X
1	0	$c_0 \|0\rangle - c_1 \|1\rangle$	Z
1	1	$c_0 \|1\rangle - c_1 \|0\rangle$	ZX

With $U^1 = U$ and $U^0 = \mathbb{1}$, the corrections can be summarized into $Z^{m_2} X^{m_1}$ which are the last two gates of circuit (2.2).

We therefore teleported a qubit from A to B! Note that this does *not* allow transmission of information faster-than-light as the two classical bits have to be transmitted. Without them, the qubit is worthless as Bob cannot interpret it correctly[1]. It also does not violate the no-cloning theorem as Alice's copy is destroyed during the measurement.

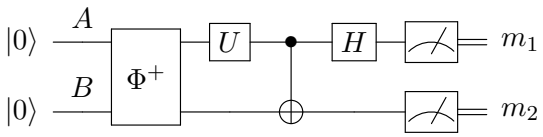
Concatenated Teleportation

2.5.3 Dense-Coding

We are now concerned with the "opposite" problem of qubit teleportation: instead of teleporting a qubit's state, we physically transport it to another location but encode two classical bits of information in it. That

[1] One might argue that Bob might get lucky and read out the correct information. But this kind of "faster-than-light transportation" is also possible classically: you can just guess what the information is—but does not actually transmit information!

is, we transmit two bits of classical information by only transmitting a single qubit. Consider the following circuit:



After creating the Bell state, Alice applies a unitary $U \in \{\mathbb{1}, X, Z, ZX\}$ and subsequently transmits the qubit to Bob. He, now in possession of both qubits A and B, now applies the rest of the gates to read out what unitary Alice applied. The measurement results are as follows:

U	m_1	m_2
$\mathbb{1}$	0	0
X	0	1
Z	1	0
ZX	1	1

Validating this is analogous to the teleportation and left as an exercise to the reader.

Again, this protocol does not allow faster-than-light communication as the qubit has to be physically transmitted. A combination with the teleportation protocol is possible, but this in turns requires the classical transmission of two bits, so still no faster-than-light transmission is possible.

2.6 Why these postulates?

One might ask *why* the postulates are as is (e.g., Why probabilities in the first place? Why amplitudes and not real positive numbers? Why the Euclidean norm and not an arbitrary p-norm? Why linearity?). The hard way to understand this is:

1. learn classical physics
2. learn why classical physics is not sufficient
3. learn quantum physics
4. maybe hear about why amplitudes and not probabilities

However, this course is not the place to squeeze in at least one year worth of lectures just to understand the postulates. Instead, we will take a more pragmatic approach, starting from why we use the Euclidean norm.

Why the Euclidean Norm? Consider $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)$ describing the probabilities of an event with N possible outcomes. We impose a condition $\|\boldsymbol{v}|_p = 1$ to ensure normalization. The most natural choice would be $p = 1$, i.e., requiring that the sum of the magnitudes is unity. However, remember that we want to apply transformations $\boldsymbol{A}$ to the vector and still keep the normalization condition: $\|\boldsymbol{v}\|_p = \|\boldsymbol{A}\boldsymbol{v}\|_p = 1$. For any p, this condition only allows permutations $v_i \mapsto v_j$ and sign flips $v_i \mapsto -v_i$. None of these are capable of encoding everything interesting! However, for $p \in \{1, 2\}$, these matrices can encode more things. For $p = 1$, stochastic matrices are allows and for $p = 2$, we can use unitary matrices! For higher p, no interesting behavior can be encoded. A very practical argument why we use $p = 2$ is therefore that otherwise QM would be very boring.

Why Complex Numbers? Again, we can bring up a very practical argument: only complex number are algebraically closed. Consider, for instance, a unitary gate U. Applying this gate takes t time. If we want to apply it for only $t/2$ time, we need to take its square-root $U = VV = V^2$. With being closed under this operation, it might be that there is no such gate! But as we are able to apply it for only $t/2$ time, there must be some form of square-root-U in the universe. Hence, we have to use complex numbers. Take, for instance, the gate $U = Z = \mathrm{diag}(1, -1) = \big(\mathrm{diag}(1, i)\big)^2$.

Why Linearity? We always have the assumption that gates progress our state linearly. If it would not, i.e., if it would progress nonlinearly, we could solve NP-complete problems! But this is unrealistic, so we confine ourselves to linear evolution…

What is Quantum Mechanics About? Quantum mechanics is not about matter, energy, waves, nor particles. Instead, its solely about information, probabilities, and observables and how these relate to each other! Whenever seeing two linear operators in QM, the sole answer to whether they commute conveys large amounts of information.

3 Computational Complexity

In this chapter we cover the basic ideas of complexity theory. As the core motivation behind QC is to speed up certain tasks, we first have to lay the ground for discussing what "speed up" actually means. In computer science, the *complex* of an algorithm describes the resources required to run it. This resource is often *space* or *time*. That is, how much memory or time it takes to run a specific algorithm. To assign a complexity to a problem instead of a specific algorithm, we assign say that the problem has the complexity of the best algorithm solving it.

The most common complexity classes are depicted in Figure 3.1. These are:

- P: problems that are solvable in polynomial time (graph connectivity, testing if a number is prime, matchmaking, sorting, linear search, ...)
- Bounded-Error Quantum Polynomial (BQP): problems solvable on a quantum computer with bounded error probability (e.g., $P(\text{error}) \leq 2/3$) (factoring, discrete logarithm, ...?)
- NP: problems believed to not be solvable in polynomial time (graph isomorphism, ...)
- NP-complete: hard problems that can be reduced on each other and for which the solution can be checked in polynomial time (box packing, map coloring, traveling salesman, $n \times n$ Sudoku, ...)
- PSPACE: problems which need polynomial amount of memory ($n \times n$ chess, $n \times n$ Go, ...)

These complexity classes are defined such that P $\subseteq$ BQP $\subseteq$ NP $\subseteq$ PSPACE. A big open problem of computer science is whether P $\neq$ NP, i.e., whether we can solve all problems "fast." A similar question comes up for QC: is P $\neq$ BQP, i.e., are there problems that can actually be solved faster on a quantum computer?

To assess the complexity of a quantum algorithm, we count the gates required to implement the circuit. We will see in the next chapter (chapter 4) how this scales with the problem size.

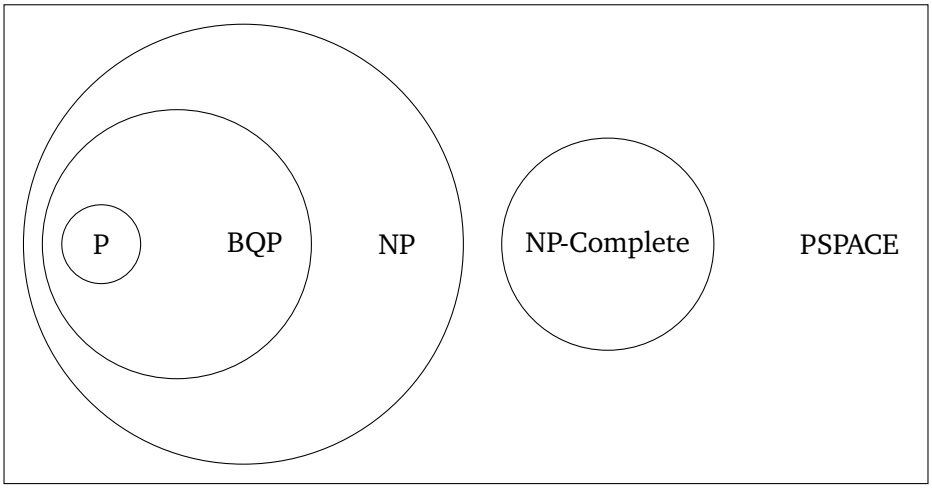
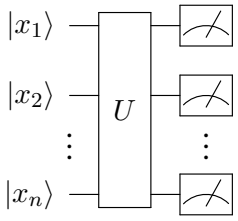


Figure 3.1: The Computational Complexity Zoo

4 Universal Computation

In general, a quantum circuit is just a unitary U and measurements,



with an initialization $|x_1 x_2 \dots x_n\rangle$. However, this circuit has to be constructed somehow from gates we have available in the lab. this raises the natural question of what gates we have to implement to build every unitary—and whether there are actually a set of gates fulfilling this.

Definition 3 (Universal Set of Gates)**.** A set of gates $\mathcal{G}$ is called *universal* if any unitary can be approximated with arbitrary accuracy using only gates from this set. With a gate U and its approximation V, let

$$E(U,V) = \max_{|\psi\rangle} \|(U-V)|\psi\rangle\|$$

be the *error* between U and V. Note that this error is an upper bound on the probability error $|P_U - P_V| \leq 2E(U,V)$ quantifying the difference in the probability distributions induced by applying U or V to a state.

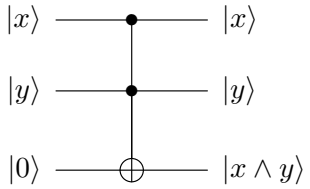
Note that is analogous to classical computing. There we could also decompose every algorithm into a set of universal logical gates. For instance, the sets $\{\,\text{AND}, \text{OR}, \text{NOT}\,\}$ and $\{\,\text{NAND}\,\}$ are both universal and can represent every possible classical circuit. We formulate this into a theorem:

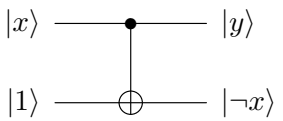
Theorem 8 (Classical Set of Universal Gates)**.** *The sets $\{\,AND, OR, NOT\,\}$ and $\{\,NAND\,\}$ and universal for all classical logic gates.*

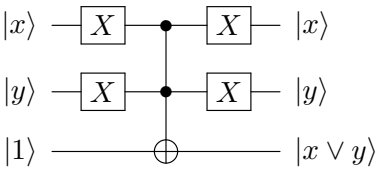
Proof. By induction. □

Theorem 9 (Embedding of Classical Circuits)**.** *Every classical circuit can be embedded in a quantum circuit performing the equivalent operation, but reversibly.*

Proof. To proof this, we use that $\{\,\text{AND}, \text{NOT}, \text{OR}\,\}$ is universal for all classical gates (Theorem 8). We therefore only have to show that these gates can be resembled using quantum circuits. For this, we use the X-, CNOT-, and Toffoli-gate:







Showing the equivalence is trivial. Note that in the logical or, the Toffoli-gate functions as a not-and due to the target qubit being set to $|1\rangle$. □

4.1 Universal Quantum Gates

In this section we will go over the proof of universality. Some common groups of quantum gates that are discussed are, for instance, the *Pauli group* $\mathcal{P} = \langle X, Z \rangle$ from which all the Pauli-gates can be constructed:

$$\langle X, Z \rangle \longrightarrow \{X^2 = Z^2 = \mathbb{1}, X, XZ = iY, ZX = -iY, Z\} \equiv \{\mathbb{1}, X, Y, Z\}.$$

Another important group is the *Clifford group* $\mathcal{C} = \langle H, S, CNOT \rangle$ with

$$\langle H, S, CNOT \rangle \longrightarrow \{H^2 = \mathbb{1}, S^2 = Z, \dots\}.$$

However, we have the following result:

Theorem 10 (Gottesman-Knill Theorem)**.** *Circuits build using solely gates from the Clifford group can be efficiently simulated on a classical computer.*

Hence, the Clifford group is not enough as we will never see a speedup when just using its gates! However, if we add the T-gate $T = \mathrm{diag}(1, e^{i\pi/4})$, we get a universal set of gates:

Theorem 11 (Universal Set of Quantum Gates)**.** *The following set of quantum gates is universal:* $\langle H, S, CNOT, T \rangle$.

Proof Sketch. The proof of this theorem has three parts:

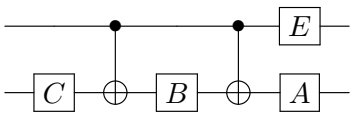
1. every unitary matrix can be decomposed into the product of two-level[1] unitary matrices
2. every two-level unitary matrix can be decomposed into CNOT- and single-qubit gates
3. every single-qubit gate can be approximated with arbitrary accuracy by $\langle H, T \rangle$

which we will cover in greater detail.

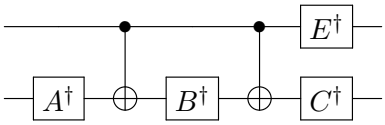
Part 1/3: For an n-qubit gate U, at most $2^{n-1}(2^n - 1) \in \mathcal{O}(4^n)$ two-level unitary matrices are needed.

Part 2/3: Let $\tilde{U} \in \mathbb{C}^{2n+1 \times 2^{n+1}}$ be a two-level unitary matrix acting on $n+1$ qubits. Assume w.l.o.g. that $\tilde{U}$ is a block diagonal matrix $\tilde{U} = \mathrm{diag}(\mathbb{1}, U)$, where $U \in \mathbb{C}^{2\times 2}$ contains the four non-trivial entries of $\tilde{U}$. Hence, $\tilde{U}$ is a the n-controlled version of U, i.e., U is only applied to the $(n+1)$-th qubit iff the first n qubits are 1. To show that this gate can be constructed using just single-qubit gates and CNOT-gates, we first construct a controlled-U-gate, then a controlled-controlled-U-gate, and subsequently expand this to an n-controlled-U-gate.

Let $U =: e^{i\alpha} AXBXC$ be a decomposition of U such that $ABC = \mathbb{1}$ (note that this is always possible due to Theorem 3). The following circuit implements a controlled-U-gate:



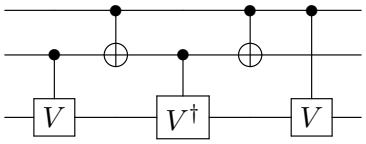
with $E := \mathrm{diag}(1, e^{i\alpha})$. For $|0x\rangle$, both E and the CNOT-gates have no effect and therefore due to $ABC = \mathbb{1}$, the state is left as is. For $|1x\rangle$, both E and the CNOT-gates are applied and therefore the unitary $CXBXA$ acts on the second qubit together with the global phase $e^{i\alpha}$, which is equivalent to applying U. Similarly,



[1] A two-level unitary matrix is a matrix that only acts non-trivially on at most two vector components.

implements the controlled-$U^\dagger$-gate necessary for the next step (the proof is analogous).

The controlled-controlled-U-gate is realized by



(4.1)

with V being the half-U-gate, i.e., $V^2 = U$. For $|00x\rangle$, no gate is ever applied. For $|01x\rangle$, V and $V^\dagger$ are applied, canceling each other out. For $|10x\rangle$, $V^\dagger$ and V are applied, canceling each other out. For $|11x\rangle$, however, both V but not $V^\dagger$ are applied (due to the CNOT-gates canceling the activation on the second qubit), resulting in $VV = V^2 = U$ being applied to the this qubit. Hence, this circuit realizes the controlled-controlled-U-gate. By some clever arrangement, this circuit needs eight single-qubit gates and six CNOT-gates.

To build the n-controlled-U-gate, we add the new control bits to the front and expand the control lines of (4.1) to these qubits, using an $(n-1)$-controlled-gate. Finally, we end up using $\mathcal{O}(n^2)$ CNOT- and single-qubit gates.

Part 3/3: By the *Solovay-Kitaev theorem,* approximating a circuit with m CNOT- and single-qubit up to an accuracy ϵ requires $\mathcal{O}\big(m \log^2(m/\epsilon)\big)$ gates from $\langle H, T\rangle$.

This concludes the proof sketch and we end up with

$$\mathcal{O}\left(4^n 2^n \log^2\left(\frac{4^n 2^n}{\epsilon}\right)\right)$$

gates to approximate an arbitrary n-qubit quantum circuit. □

From this discussion and the final gate count that scales exponentially with the number of qubit, it does not appear clear why anyone should think that BQP $\neq$ P. Even classical circuits need an exponential amount of time on a quantum computer! This is the reason why algorithms that are efficient on a quantum computer are rare and require a large amount of creativity. The next chapter covers nearly all quantum algorithms that we know so far, which only reinforces the argument how much creativity is necessary to invent new ones.

5 Algorithms

After discussing basic protocols and results in section 2.5, we will now go over actual algorithms. After a discussion of the main idea underlying all algorithms, *quantum parallelism,* in section 5.1, we will cover the following algorithms:

- Deutsch-Josza Algorithm (section 5.2)
  Finds out whether a function is constant or balanced. Main purpose is to show quantum speedup compared to deterministic classical computing. Exponential speedup, but efficiently solvable on a probabilistic classical computer.
- Bernstein-Vazirani Algorithm (section 5.3)
  Finds a binary string hidden in a special function. Main purpose is to show quantum speedup compared to probabilistic classical computing. Linear speedup.
- Simon's Algorithm (section 5.4)
  Finds the periodicity of a 2-to-1-function. Exponential speedup!
- Quantum Fourier Transform (section 5.5)
  Finds the periods hidden in data; quantum version of the famous Fourier transform. Exponential speedup!
- Shor's Algorithm (section 5.6)
  Factors two large numbers; can be used to break RSA encryption. Almost exponential speedup.
- Grover's Algorithm (section 5.7)
  Finds a value in an unstructured[1] pool of values. Square-root speedup.

A brief overview over the most important aspects of each algorithm is given in Table 5.1, but I recommend to first work through this chapter and then take a look at the table.

[1] Here, *unstructured* refers to the search space which has no special structure, e.g., being sorted which would allow efficient classical algorithms such as binary search.

Name	Tackled Problem	Classical Complexity	Quantum Complexity
Deutsch-Josza Algorithm	Deutsch's Problem	$\mathcal{O}(2^n)$	$\mathcal{O}(1)$
Bernstein-Vazirani Algorithm	Hidden String	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Simon's Algorithm	Simon's Problem	$\mathcal{O}(2^n)$	$\mathcal{O}(n)$
Quantum Fourier Transform	Fourier Transform	$\mathcal{O}(2^n)$	$\mathcal{O}(n^2)$
Shor's Algorithm	Factoring	$\mathcal{O}(e^{\log N})$	$\mathcal{O}\big((\log N)^3\big)$
Grover's Algorithm	Unstructured Search	$\mathcal{O}(N-M)$	$\mathcal{O}(\sqrt{N/M})$

Table 5.1: Quantum Algorithms

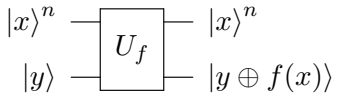
5.1 Quantum Parallelism

In chapter 4, we saw that it is possible to implement arbitrary circuits on a quantum computer. However, exponentially many gates where necessary! So what is the reason why we think that QC accelerates computation? Classically, we can evaluate some function only one input at a time. In QC, however, we can make use of the linear evolution. We can therefore prepare a superposition, apply the query unitary, and therefore evaluate it on all 2^n inputs simultaneously. For brevity, we denote quantum wires containing more than one qubit as

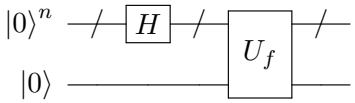
or will simply keep the multi-qubit wire implicit. However, we can not use the calculation of this circuit as we do not know which input produced which output! We therefore have to come up with a better approach based on *interference*.

5.1.1 Interference and Deutsch's Approach

Consider a function $f : \{0,1\}^n \to \{0,1\} : x \mapsto f(x)$ with the unitary $U_f : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$ (we will proof that this is in fact a unitary transformation in subsection 5.1.2). In circuit notation, we write:



This circuit keeps the inputs and the outputs together! We can now make use of superposition,

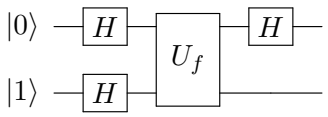


which produces the following output:

$$|0\rangle^n |0\rangle \xrightarrow{H_{1:n}} \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n} \otimes |0\rangle \xrightarrow{U_f} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle |f(x)\rangle$$

We have now evaluated f with just one call for all possible inputs! However, we cannot read all the results as a measurement would collapse the superposition onto just one evaluation and we could have just called the classical implementation. Hence, we need a slightly different approach to quantum parallelism.

Assume $n = 1$, i.e., a single input qubit. Then *Deutsch's approach* is to also use a superposition as the "input" for the target qubit and to apply a second Hadamard gate after the unitary on the input qubit:



(5.1)

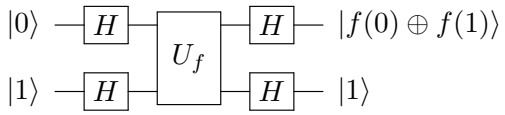
The core idea is now how U_f acts an the product state of a basis state $|x\rangle$ and $|-\rangle$:

$$\begin{aligned} U_f\left(|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) &= |x\rangle \otimes \frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) \\ &= |x\rangle \otimes \frac{1}{\sqrt{2}} \begin{cases} |0\rangle - |1\rangle & \text{if } f(x) = 0 \\ |1\rangle - |0\rangle & \text{if } f(x) = 1 \end{cases} \\ &= (-1)^{f(x)} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

This shows that we can pull the result of $f(x)$ from the second qubit and put it to a global phase! We can now continue the evaluation of (5.1):

$$\begin{aligned}
(H\otimes\mathbb{1})U_f(H\otimes H)\left|0\right\rangle\left|1\right\rangle &= (H\otimes\mathbb{1})U_f\left[\frac{1}{\sqrt{2}}(\left|0\right\rangle+\left|1\right\rangle)\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}\right]\\
&= (H\otimes\mathbb{1})\frac{1}{\sqrt{2}}U_f\left[\left|0\right\rangle\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}+\left|1\right\rangle\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}\right]\\
&= (H\otimes\mathbb{1})\frac{1}{\sqrt{2}}\left[(-1)^{f(0)}\left|0\right\rangle\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}+(-1)^{f(1)}\left|1\right\rangle\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}\right]\\
&= (H\otimes\mathbb{1})\frac{1}{\sqrt{2}}\left[(-1)^{f(0)}\left|0\right\rangle+(-1)^{f(1)}\left|1\right\rangle\right]\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}\\
&= (H\otimes\mathbb{1})\frac{1}{\sqrt{2}}\begin{cases}\pm(\left|0\right\rangle+\left|1\right\rangle) & \text{if } f(0)=f(1)\\ \pm(\left|0\right\rangle-\left|1\right\rangle) & \text{if } f(0)\neq f(1)\end{cases}\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}\\
&= \begin{cases}\pm\left|0\right\rangle & \text{if } f(0)=f(1)\\ \pm\left|1\right\rangle & \text{if } f(0)\neq f(1)\end{cases}\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}\\
&= \pm\left|f(0)\oplus f(1)\right\rangle\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}\\
&\equiv \left|f(0)\oplus f(1)\right\rangle\otimes\frac{(\left|0\right\rangle-\left|1\right\rangle)}{\sqrt{2}}
\end{aligned}$$

Finally, we can measure the first qubit. But we did still not read out evaluations of f! It turns out that in QC, we can often not read out individual results, but global properties of functions. In this case, a measurement 0 means that our function is constant and 1 that it is not. Also, we did not use the target qubit in the end—it was only useful for transferring function results into the global phase. In fact, we usually apply a Hadamard gate to it in the end to restore the original state. Hence, the entire "algorithm" of Deutsch's approach is:



(5.2)

This approach uses quantum parallelism and interference to learn a global property of f. All subsequent algorithms will rely on this or a similar technique for their speedup along with massive classical post-processing in some cases.

5.1.2 The Query Unitary

So far, we have just assumed that $U_f:\left|x\right\rangle\left|y\right\rangle\mapsto\left|x\right\rangle\left|y\oplus f(x)\right\rangle$ is a unitary transformation for any function f. In this section we will see that this is actually the case and how U_f looks.

Theorem 12 (Query Unitary). *Let $f:\{0,1\}^n\to\{0,1\}:x\mapsto f(x)$ be some function. Then the mapping $U_f:\left|x\right\rangle\left|y\right\rangle\mapsto\left|x\right\rangle\left|y\oplus f(x)\right\rangle$ is unitary.*

Proof. We proof this by constructing U_f. Let $i,k\in\{0,1\}^n$ and $j,\ell\in\{0,1\}$, then the $(ij,k\ell)$-th matrix element of U_f is

$$\left\langle j\right|\left\langle i\right|U_f\left|k\right\rangle\left|\ell\right\rangle=\left\langle j\right|\left\langle i\right|\left(U_f\left|k\right\rangle\left|\ell\right\rangle\right)=\left\langle j\right|\left\langle i\right|\left(\left|k\right\rangle\left|\ell\oplus f(k)\right\rangle\right)=\left\langle i|k\right\rangle\left\langle j|\ell\oplus f(k)\right\rangle=\delta_{i,k}\delta_{j,\ell\oplus f(k)}.$$

From this we see that the matrix U_f is block-diagonal w.r.t. the first n qubits. The ik-th block (which we can also call the i-th or k-th block as the matrix is diagonal) is then

$$(U_f)_{ik} = \begin{bmatrix} \delta_{0,0\oplus f(k)} & \delta_{0,1\oplus f(k)} \\ \delta_{1,0\oplus f(k)} & \delta_{1,1\oplus f(k)} \end{bmatrix} = \begin{bmatrix} \delta_{0,f(k)} & \delta_{0,1\oplus f(k)} \\ \delta_{1,f(k)} & \delta_{1,1\oplus f(k)} \end{bmatrix} = \begin{bmatrix} \delta_{0,f(k)} & \delta_{1,f(k)} \\ \delta_{1,f(k)} & \delta_{0,f(k)} \end{bmatrix} \doteq \begin{bmatrix} m_k & 1\oplus m_k \\ 1\oplus m_k & m_k \end{bmatrix}$$

with $m_k \coloneqq \delta_{0,f(k)}$. As m_k is constant within the k-th block, it is either $\mathbb{1}$ or X for $f(k) = 0$ or $f(k) = 1$, respectively. As a block-diagonal matrix of unitary matrices is still unitary, U_f is unitary for every f. □

5.2 Deutsch-Josza Algorithm

In this section we explore the *Deutsch-Josza algorithm* used to solve *Deutsch's problem*. We first describe the problem, then the classical solution, and finally the quantum approach. Alongside we discuss complexity arguments and run-time.

5.2.1 Problem

Assume a function $f : \{0,1\}^n \to \{0,1\}$ that is either *constant* or *balanced,* i.e., $f(x) = a$ for all x or $f(x) = 0$ for half x and $f(x) = 1$ for the other half. We want to find out which of these cases f falls into, i.e., whether f is constant or balanced. This problem is also called *Deutsch's problem*.

5.2.2 Classical Approach

To determine with certainty whether a given function (of n bits) is constant or balanced, we need to evaluate it for one more than half the inputs. That is, $2^{n-1}+1$ times. The classical algorithm is therefore in $\mathcal{O}(2^n)$.

5.2.3 Quantum Approach

For solving Deutsch's problem in a quantum way, we directly apply Deutsch's approach (5.2) with a unitary U_f implementing f. This yields the following state (while dropping the last qubit as its state does not convey information) before the final Hadamard-layer:

$$\frac{1}{\sqrt{2^n}} H^{\otimes n} \sum_x (-1)^{f(x)} |x\rangle \tag{5.3}$$

We now have two cases: either f is constant or f is balanced. We first consider the first (constant) case. Then the above state reduces to

$$\frac{1}{\sqrt{2^n}} H^{\otimes n} \sum_x (-1)^{f(x)} |x\rangle = \pm \frac{1}{\sqrt{2^n}} H^{\otimes n} \sum_x |x\rangle = \pm H^{\otimes n} |+\rangle^{\otimes n} = \pm |0\rangle^{\otimes n} \equiv |0\rangle^{\otimes n}.$$

We now turn to the second case, a balanced f. For this, we separate the x's into x' and x'' which are the values for which $f(x') = 0$ and $f(x'') = 1$, respectively. Using (4), we can write the state (5.3) as

$$\begin{aligned}
\frac{1}{\sqrt{2^n}} H^{\otimes n} \sum_x (-1)^{f(x)} |x\rangle &= \frac{1}{\sqrt{2^n}} H^{\otimes n} \left[\sum_{x'} |x'\rangle - \sum_{x''} |x''\rangle \right] \\
&= \frac{1}{2^n} \left[\sum_{x'} \sum_y (-1)^{x' \cdot y} |y\rangle - \sum_{x''} \sum_y (-1)^{x'' \cdot y} |y\rangle \right] \\
&= \frac{1}{2^n} \sum_y \left[\sum_{x'} (-1)^{x' \cdot y} - \sum_{x''} (-1)^{x'' \cdot y} \right] |y\rangle \\
&= \frac{1}{2^n} \left(\left[\sum_{x'} (-1)^0 |y\rangle - \sum_{x''} (-1)^0 |y\rangle \right] + \sum_{y \neq 0} \left[\sum_{x'} (-1)^{x' \cdot y} - \sum_{x''} (-1)^{x'' \cdot y} \right] |y\rangle \right) \\
&= \frac{1}{2^n} \left(\left[\sum_{x'} |y\rangle - \sum_{x''} |y\rangle \right] + \sum_{y \neq 0} \left[\sum_{x'} (-1)^{x' \cdot y} - \sum_{x''} (-1)^{x'' \cdot y} \right] |y\rangle \right) \\
&= \frac{1}{2^n} \sum_{y \neq 0} \left[\sum_{x'} (-1)^{x' \cdot y} - \sum_{x''} (-1)^{x'' \cdot y} \right] |y\rangle
\end{aligned}$$

and remove $|0\rangle^{\otimes n}$! Hence, it is impossible to measure all zeros for a balanced function and we can therefore deduce the following rules:

- if all measurement outcomes are zero, f is constant
- if any measurement outcome is one, f is balanced

Opposed to the classical algorithm which needed $2^{n-1} + 1$ calls of f, the Deutsch-Josza algorithm only needed a single call and is therefore in $\mathcal{O}(1)$.

5.2.4 Remarks

We saw that the Deutsch-Josza algorithm provides an exponential speedup over the classical solution. This gives a good start for QC and shows that we may actually be faster than classical computing given that we come up with a clever algorithm. However, it turns out that Deutsch's problem as absolutely no real applications and that probabilistic classical computers can also do a pretty good job at solving it.

The algorithm is summarized in algorithm 1.

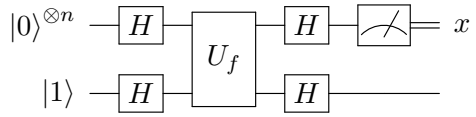
5.2.5 Modified Deutsch-Josza Algorithm

5.3 Bernstein-Vazirani Algorithm

In this section we explore the *Bernstein-Vazirani algorithm* used to extract a hidden string from a function. We first describe the problem, then the classical solution, and finally the quantum approach. Alongside we discuss complexity arguments and run-time.

Algorithm 1: Deutsch-Josza Algorithm

Input: constant or balanced function $f : \{0,1\}^n \to \{0,1\}$
Output: whether the function is constant or balanced
// Run quantum circuit.

1 

2 **if** *all* x *are zero* **then**
3 **return** "constant"
4 **return** "balanced"

5.3.1 Problem

Assume a function $f : \{0,1\}^n \to \{0,1\}$ that is either *constant* or *balanced,* i.e., $f(x) = a$ for all x or $f(x) = 0$ for half x and $f(x) = 1$ for the other half. We want to find out which of these cases f falls into, i.e., whether f is constant or balanced. This problem is also called *Deutsch's problem.*

Assume a function $f : \{0,1\}^n \to \{0,1\}$ that hides a bit string $s \in \{0,1\}^n$ as follows:

$$f(x) = x \cdot s = x_1 s_1 \oplus x_2 s_2 \oplus \cdots \oplus x_n s_n.$$

The goal of the problem is to find s with as few calls of f as possible.

5.3.2 Classical Approach

Classically, we would need n evaluations of f on the basis vectors, i.e., $f(e_j) = s_j$ and the classical solution is therefore $\mathcal{O}(n)$.

5.3.3 Quantum Approach

For the quantum approach, we again directly apply Deutsch's approach (5.2) with a unitary U_f implementing f. Before the final Hadamard-layer, we now have the following state (dropping the last qubit as it does not convey any information):

$$\frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle$$

As before, it boils down to cleverly manipulating the state to find an interpretation of the measurement outcomes. We now plug in the definition of f (and replace the module-two-sum with an actual sum as we are only using the result in an exponent of -1). By cleverly using power rules and rearranging sum and tensor product, we find that the state is:

$$\begin{aligned}
\frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle &= \frac{1}{\sqrt{2^n}} \sum_x (-1)^{\sum_{i=1}^n x_i s_i} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_x \bigotimes_{i=1}^n (-1)^{x_i s_i} |x_i\rangle_i \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{i=1}^n \sum_{x_i} (-1)^{x_i s_i} |x_i\rangle_i = \frac{1}{\sqrt{2^n}} \bigotimes_{i=1}^n |0\rangle_i + (-1)^{s_i} |1\rangle_i = \bigotimes_{i=1}^n \begin{cases} |+\rangle_i & \text{if } s_i = 0 \\ |-\rangle_i & \text{if } s_i = 1 \end{cases}
\end{aligned}$$

And if we apply the Hadamard-gate again, we simply get the state $|s\rangle$! Measuring therefore reveals s with certainty. Much like in the Deutsch-Josza algorithm, we only need a single invocation of U_f and therefore this algorithm is also in $\mathcal{O}(1)$.

5.3.4 Remarks

We saw that the Bernstein-Vazirani algorithm provides a linear speedup over the classical solution. While the applications of this algorithm are still fairly limited, it gives rise to some ideas that will become useful later on. Also, the hidden string problem is not efficiently solvable on classical computers, also probabilistic classical computers!

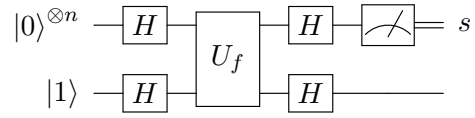
The algorithm is summarized in algorithm 2.

Algorithm 2: Bernstein-Vazirani Algorithm

Input: function $f : \{0,1\}^n \to \{0,1\} : x \mapsto x \cdot s$ with a hidden string $s \in \{0,1\}^n$

Output: the hidden string s

// Run quantum circuit.

1 

2 **return** s

5.4 Simon's Algorithm

From this section on, the algorithms become more involved. Most notable, we will need classical post-processing to make sense of the measurements. The first algorithm of this kind is Simon's algorithm which solves Simon's problem (which, as usual, will be introduced before looking at any approaches). Also, it is the first algorithm that gives us the solution only up to some probability.

5.4.1 Problem

In *Simon's problem* we again try to find a string $a \in \{0,1\}^n$ hidden in a function $f : \{0,1\}^n \to \{0,1\}^n$. However, we assume that f is a two-to-one-function such that $f(x) = f(y)$ iff $x = y \oplus a$ where $\oplus$ denotes bit-wise addition. We are therefore trying to find a period a of a function f under bit-wise addition such that $f(x \oplus a) = f(x)$.

5.4.2 Classical Approach

Classically, we evaluate f at various different inputs and compare the result to the previous results. If we find a match, we can directly compute the period a. If we do not find a match on the j-th evaluation, we have eliminated

$$1 + 2 + \cdots + (j-1) = j(j-1)/2$$

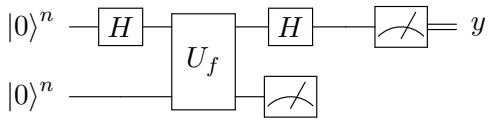
possible values of a. This is due to our choice of the inputs: we always choose different values and compare the output to all lower. In the first iteration, we cannot compare to anything. In the second, we eliminate one a. In the third, we eliminate two others, and so on. As there are $2^n - 1$ possible choices of a, we do not have an acceptable probability of finding a as long as $j(j-1)/2 \ll 2^n$. Hence, we need around $j = \sqrt{2^n}$ iterations! Therefore, the classical approach is in $\mathcal{O}(\sqrt{2^n})$.

5.4.3 Quantum Approach

As mentioned already is the quantum approach to Simon's problem—Simon's algorithm—more complicated than both Deutsch-Josza and Bernstein-Vazirani and requires classical post-processing. We therefore split up this section into the quantum circuit and the classical post-processing.

Circuit

Opposed to the previous two algorithms, we do not directly apply Deutsch's approach. Instead, we only put the first n qubits into a superposition and use zero-qubits for the targets:



After the U_f-gate, we have the state

$$|0\rangle^n |0\rangle^n \xrightarrow{H^{\otimes n}} |+\rangle^n |0\rangle^n = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle |0\rangle^n \xrightarrow{U_f} \frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_i} \frac{|x_i\rangle + |x_i \oplus a\rangle}{2} |f(x_i)\rangle$$

where the sums over x are over all n-bit binary strings and the last sum (over x_i) is over the individual values producing distinct values $f(x_i)$. As f is a two-to-one function where $f(x) = f(x \oplus a)$ for a constant a, we can pull in this second value explicitly by adding a. By now measuring the target qubits (the ones storing the result of f), we collapse the first n qubits onto a state containing only one x_i and its "periodic equivalent":

$$\frac{1}{\sqrt{2}}(|x_i\rangle + |x_i \oplus a\rangle).$$

Note that while this seems great at a first glance, we cannot directly extract a from this state as we can only measure once, giving us either x_i or $x_i \oplus a$, and we cannot guarantee to end up in this state again by just repeating this experiment (as we might collapse onto a different $f(x)$). By no-cloning, we can also not just copy the state and just repeating until we get the same outcome again would reduce the quantum approach to the classical approach, but with the overhead of buying a quantum computer.

We therefore have to do some more work. Using the idea of Deutsch's approach again (interference) and applying H with Theorem 4, we can pull the computation results into a phase (we drop the index $\cdot_i$ here for brevity) and remove x from the parts "that matter":

$$\begin{aligned}
\frac{1}{\sqrt{2}} H^{\otimes n}(|x_i\rangle + |x_i \oplus a\rangle) &= \frac{1}{\sqrt{2^{n+1}}} \sum_y \left[(-1)^{x \cdot y} + (-1)^{(x \oplus a) \cdot y}\right] |y\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_y \left[(-1)^{x \cdot y} + (-1)^{x \cdot y}(-1)^{a \cdot y}\right] |y\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_y (-1)^{x \cdot y} \left[1 + (-1)^{a \cdot y}\right] |y\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_y (-1)^{x \cdot y} \delta_{a \cdot y, 0} |y\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{y,\, a \cdot y = 0} (-1)^{x \cdot y} |y\rangle
\end{aligned}$$

After applying the Hadamard gate, we are only left with an equal superposition of y's that have the property of $a \cdot y = 0$ which there are 2^{n-1} many. We quickly want to give an argument why thus number is correct. Assume

w.l.o.g.[2] that the last bit of a is one, i.e., a has the form $a = a_1 a_2 \dots a_{n-1} 1$ with arbitrary $a_1, a_2, \dots, a_{n-1}$. Then the first $n-1$ bits of $y = y_1 y_2 \dots y_{n-1} y_n$ can be freely chosen from $\{0,1\}$ and the last bit has to be $y_n = a_1 y_1 \oplus a_2 y_2 \oplus \dots \oplus a_{n-1} y_{n-1}$. Therefore, we have 2^{n-1} possible y's.

Measuring the first n qubits at this point yields us one y that fulfills the constraint $a \cdot y = 0$. By repeating this procedure m times, we get m constraint on a and what remains is some classical post-processing to determine a.

Post-Processing

As we saw in the previous section, after the quantum part we end up with m constraints

$$a \cdot y_1 = 0 \qquad a \cdot y_2 = 0 \qquad \cdots \qquad a \cdot y_m = 0$$

on the period a. However, if y_i is zero, we do not learn anything about a and if $y_i = y_j$ for $i \neq j$, we also do not learn anything new. But as there are exponentially many y', we have[3]

$$P(y_i = 0 \vee y_i = y_j) = \frac{1}{2^{n-1}} \approx 0.$$

Hence, we have a high chance of learning something about a with each measurement and every time we learn something, we *half* the number of possible a's (cf. classically where we only get a linear reduction) and we need n distinct non-zero values to uniquely determine a. Overall, we have the following theorem:

Theorem 13 (Simon's Algorithm)**.** *The probability p of acquiring enough information to uniquely determine a with $n+m$ invocations satisfies*

$$p > 1 - \frac{1}{2^{m+1}}.$$

Proof. See Mermin, 2016, Appendix G. □

Hence, with $m = 20$, the odds of finding a are one to more than a million. Hence, Simon's algorithm is in $\mathcal{O}(n)$.

5.4.4 Remarks

We saw that Simon's algorithm provides an exponential speedup over the classical solution. And while applications of Simon's problem are rare, its main contribution to QC was motivating the ideas behind *Shor's algorithm* which has very practical applications and which we will discuss in section 5.6 after covering quantum Fourier transform (QFT) in section 5.5.

The algorithm is summarized in algorithm 3.

5.5 Quantum Fourier Transform

For Shor's algorithm (section 5.6), we will need a quantum version of the famous Fourier transform, *quantum Fourier transform (QFT)*. in this section we will see how we can implement this transform and what it complexity is compared to the classical Fourier transform. We define the QFT as follows:

[2] As $a \neq 0$, at least one bit must be one and we can just reorder our bits.

[3] Note that with a similar probability we might get two measurement outcomes during the first measurement that equal each other and with a $50\,\%$ chance we get the second value of y and can instantly deduce a. But this probability is *very* low.

Algorithm 3: Simon's Algorithm

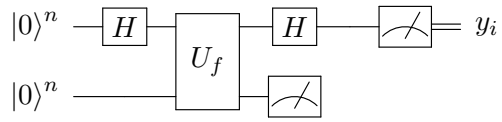
Input: two-to-one function $f : \{0,1\}^n \to \{0,1\}^n$ with the property $f(x) = f(x \oplus a)$ for a constant period $a \in \{0,1\}^n$; number of additional calls m

Output: the period a

$\mathcal{Y} \leftarrow \{\}$ // Prepare the sample set.

foreach $i = 1, 2, \ldots, n+m$ **do**

// Run quantum circuit.



// Rule out uninforming zeros and duplicates.

if $y_i \neq 0$ *and* $y_i \notin \mathcal{Y}$ **then**

$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y_i\}$ // Add "meaningful" sample to the sample set.

$a^* \leftarrow \text{solve}_a\{a \cdot y_i \mid i = 1, 2, \ldots, n+m\}$ // Gauss-Jordan Elimination

return a^*

Definition 4 (Quantum Fourier Transform)**.** The *quantum Fourier transform* of an arbitrary state $|j\rangle$ of an orthonormal basis $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$ is a linear transformation such that

$$|j\rangle \xrightarrow{U_{\text{FT}}} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n} |k\rangle \tag{5.4}$$

where j and k are the decimal representations of their binary counterpart, i.e., "actual" numbers. On an arbitrary state $|x\rangle$, the action is

$$\sum_{j=0}^{2^n-1} x_j |j\rangle \xrightarrow{U_{\text{FT}}} \sum_{k=0}^{2^n-1} y_k |k\rangle \quad \text{with} \quad y_k = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} x_j e^{2\pi i jk/2^n}.$$

Before implementing a circuit implementing QFT, we first want to show that the operator is actually unitary:

Theorem 14. *The quantum Fourier transform as defined in 4 is a unitary transformation.*

Proof. Let $|\alpha\rangle$ and $|\beta\rangle$ be arbitrary basis states (of the same basis). To show that U_{FT} is a unitary transformation, it is enough to show that the transformed basis states are still orthonormal. This is due to

$$\langle\alpha| U_{\text{FT}}^\dagger U_{\text{FT}} |\beta\rangle = \langle\alpha| \underbrace{U_{\text{FT}}^\dagger U_{\text{FT}}}_{=\mathbb{1}} |\beta\rangle = \langle\alpha|\beta\rangle = \delta_{\alpha\beta}$$

and $U_{\text{FT}}^\dagger U_{\text{FT}} = \mathbb{1}$ holding if and only if U_{FT} is unitary. We proceed by calculating the first inner product (and applying U_{FT} according to its definition):

$$\begin{aligned}
\langle\alpha| U_{\text{FT}}^\dagger U_{\text{FT}} |\beta\rangle &= \frac{1}{2^n} \left(\sum_{k_1=0}^{2^n-1} e^{-2\pi i \alpha k_1/2^n} \langle k_1| \right) \left(\sum_{k_2=0}^{2^n-1} e^{2\pi i \beta k_2/2^n} |k_2\rangle \right) \\
&= \frac{1}{2^n} \sum_{k_1=0}^{2^n-1} \sum_{k_2=0}^{2^n-1} e^{-2\pi i \alpha k_1/2^n} e^{2\pi i \beta k_2/2^n} \langle k_1|k_2\rangle = \frac{1}{2^n} \sum_{k_1=0}^{2^n-1} \sum_{k_2=0}^{2^n-1} e^{-2\pi i \alpha k_1/2^n} e^{2\pi i \beta k_2/2^n} \delta_{k_1 k_2} \\
&= \frac{1}{2^n} \sum_{k=0}^{2^n-1} e^{-2\pi i \alpha k/2^n} e^{2\pi i \beta k/2^n} = \frac{1}{2^n} \sum_{k=0}^{2^n-1} e^{2\pi i (\alpha-\beta) k/2^n}
\end{aligned}$$

We now have to continue the proof piecewise. For $\alpha = \beta$, the inner product is obviously 1. For $\alpha \neq \beta$, the sum is a special case of Theorem 1 with $a \triangleq \alpha - \beta$ and $n \triangleq 2^n$ and therefore it is equal to zero. Hence, the basis states keep their orthonormality and U_{FT} is indeed a unitary transformation. □

We now continue with building the circuit implementing QFT.

Note that this "algorithm" is different from the previous ones in that we do not go over the classical discrete Fourier transform. We will just use the result of the QFT during the discussion of Shor's algorithm.

5.5.1 Binary Fraction Expansion

To build the QFT circuit, we first rewrite (5.4) a bit using the notation $|j\rangle = |j_1 j_2 \dots j_n\rangle$ with $j = \sum_{i=1}^{n} j_i 2^{n-i}$ and the recursively defined *binary fraction*

$$0.j \coloneqq \frac{j}{2} \qquad\qquad 0.j_1 j_2 j_3 \dots j_n \coloneqq \frac{j_1}{2} + \frac{1}{2} 0.j_2 j_3 \dots j_n \tag{5.5}$$

which can also be written as

$$0.j_1 j_2 \dots j_n = \sum_{i=1}^{n} \frac{j_i}{2^i}.$$

The that it is the *order* and not the *indices* of the arguments that matter here! This is why the recursive definition is clearer. With these notations at hand, we have the following result:

Theorem 15 (Binary Fraction Expansion of QFT)**.** *The QFT result (4) can be expressed as follows:*

$$|j_1 j_2 \dots j_n\rangle \xrightarrow{U_{\mathrm{FT}}} \frac{1}{\sqrt{2^n}} \Big(\big(|0\rangle + e^{2\pi i 0.j_n} |1\rangle\big) \otimes \big(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle\big) \otimes \dots \otimes \big(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n} |1\rangle\big) \Big)$$

using binary fractions (5.5).

Proof. As usual, we show this equivalence by clever manipulation of the state. For brevity, we drop the normalization constant as it stays the same anyway. By rearranging and using the definition of exponents, we have

$$\begin{aligned} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} |k\rangle &= \sum_{k_1} \sum_{k_2} \cdots \sum_{k_n} e^{2\pi i j \sum_{\ell=1}^{n} k_\ell 2^{n-\ell} / 2^n} |k_1 k_2 \dots k_n\rangle \\ &= \sum_{k_1} \sum_{k_2} \cdots \sum_{k_n} e^{2\pi i j \sum_{\ell=1}^{n} k_\ell 2^{-\ell}} |k_1 k_2 \dots k_n\rangle = \sum_{k_1} \sum_{k_2} \cdots \sum_{k_n} \bigotimes_{\ell=1}^{n} e^{2\pi i j k_\ell 2^{-\ell}} |k_\ell\rangle =: (*). \end{aligned}$$

We now pull the sums into the tensor product one by one and add them explicitly:

$$\begin{aligned}
(*) &= \sum_{k_1}\sum_{k_2}\cdots\sum_{k_{n-1}}\bigotimes_{\ell=1}^{n} e^{2\pi i j k_\ell 2^{-\ell}}\,|k_\ell\rangle \otimes \left[\sum_{k_n} e^{2\pi i j k_n/2^n}\,|k_n\rangle\right] \\
&= \sum_{k_1}\sum_{k_2}\cdots\sum_{k_{n-1}}\bigotimes_{\ell=1}^{n} e^{2\pi i j k_\ell 2^{-\ell}}\,|k_\ell\rangle \otimes \left[e^{2\pi i j\cdot 0/2^n}\,|0\rangle + e^{2\pi i j\cdot 1/2^n}\,|1\rangle\right] \\
&= \sum_{k_1}\sum_{k_2}\cdots\sum_{k_{n-1}}\bigotimes_{\ell=1}^{n} e^{2\pi i j k_\ell 2^{-\ell}}\,|k_\ell\rangle \otimes \left[|0\rangle + e^{2\pi i j/2^n}\,|1\rangle\right] \\
&= \sum_{k_1}\sum_{k_2}\cdots\sum_{k_{n-2}}\bigotimes_{\ell=1}^{n} e^{2\pi i j k_\ell 2^{-\ell}}\,|k_\ell\rangle \otimes \left[|0\rangle + e^{2\pi i j/2^{n-1}}\,|1\rangle\right] \otimes \left[|0\rangle + e^{2\pi i j/2^n}\,|1\rangle\right] \\
&= \cdots \\
&= \bigoplus_{\ell=1}^{n}\left[|0\rangle + \underbrace{e^{2\pi i j 2^{-\ell}}}_{\alpha_\ell :=}\,|1\rangle\right]
\end{aligned}$$

But this is till not the form we desire! We have to rearrange phases of the $|1\rangle$'s more:

$$\alpha_\ell = e^{2\pi i j 2^{-\ell}} = e^{2\pi i \sum_{m=1}^{n} j_m 2^{n-m} 2^{-\ell}} = e^{2\pi i \sum_{m=1}^{n} j_m 2^{n-m-\ell}} = \prod_{m=1}^{n} e^{2\pi i j_m 2^{n-\ell-m}}$$

In this product, however, all terms with $m \leq n-\ell$ are 1 as $2^{n-\ell-m} \in \mathbb{N}_0$ for $m \leq n-\ell$ and therefore they disappear from the product:

$$\prod_{m=1}^{n} e^{2\pi i j_m 2^{n-\ell-m}} = \prod_{m=n-\ell+1}^{n} e^{2\pi i j_m 2^{n-\ell-m}} = e^{2\pi i \sum_{m=n-\ell+1}^{n} j_m 2^{n-\ell-m}} \doteq e^{2\pi i 0.j_{n-\ell+1} j_{n-\ell+2}\cdots j_n}$$

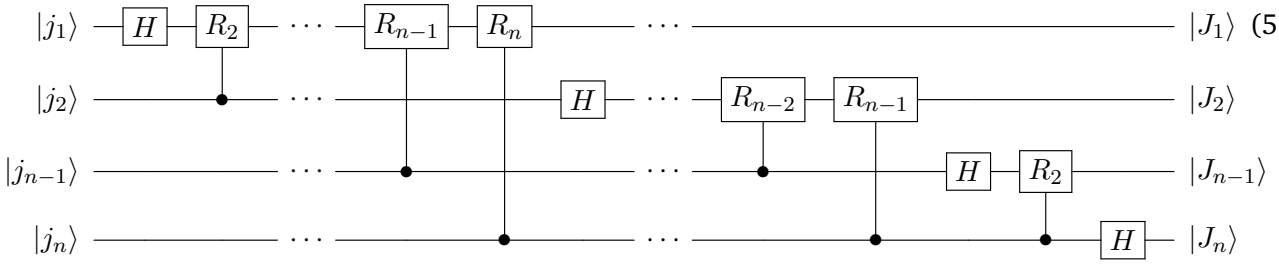
As the last equivalence is a bit hard to see, consider $\ell = 1$, $\ell = 2$, and $\ell = n$ explicitly:

- $\sum_{m=n-1+1}^{n} j_m 2^{n-1-m} = \sum_{m=n}^{n} j_m 2^{n-1-m} = j_n/2 = 0.j_n$
- $\sum_{m=n-2+1}^{n} j_m 2^{n-2-m} = \sum_{m=n-1}^{n} j_m 2^{n-2-m} = j_{n-1}/2 + j_n/4$
- $\sum_{m=n-n+1}^{n} j_m 2^{n-n-m} = \sum_{m=1}^{n} j_m 2^{-m} = j_1/2 + j_2/4 + \cdots + j_n/2^n$

Plugging everything together, we have the result that we wanted to show. □

5.5.2 Quantum Circuit

As the QFT is not simply an application of a unitary (or at least we do not want to use a bruteforce approach to generate it), we need to consider the qubits individually. First, we show that



(5.6)

produces the following states (with $R_k \coloneqq \mathrm{diag}(1, e^{2\pi i/2^k})$):

$$\begin{aligned} |J_1\rangle &= \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_{n-1} j_n}\,|1\rangle\big) & |J_2\rangle &= \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_2 \dots j_{n-1} j_n}\,|1\rangle\big) \\ |J_{n-1}\rangle &= \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}\,|1\rangle\big) & |J_n\rangle &= \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_n}\,|1\rangle\big) \end{aligned} \tag{5.7}$$

For the first qubit, we have

$$\begin{aligned} |j_1 j_2 \dots j_{n-1} j_n\rangle &\xrightarrow{H_1} \frac{1}{\sqrt{2}}\big(|0\rangle + \underbrace{(-1)^{j_1}}_{=e^{\pi i j_1} = e^{2\pi i j_1/2}}\,|1\rangle\big)\,|j_2 \dots j_{n-1} j_n\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_1}\,|1\rangle\big)\,|j_2 \dots j_{n-1} j_n\rangle \\ &\xrightarrow{CR_2} \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_1} e^{2\pi i j_2/4}\,|1\rangle\big)\,|j_2 \dots j_{n-1} j_n\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_1 j_2}\,|1\rangle\big)\,|j_2 \dots j_{n-1} j_n\rangle \\ &\longrightarrow \dots \\ &\xrightarrow{CR_n} \frac{1}{\sqrt{2}}\big(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}\,|1\rangle\big)\,|j_2 \dots j_{n-1} j_n\rangle\,. \end{aligned}$$

As the result of the second qubit is independent of the first qubit and so on (i.e., the i-th qubit does not effect the j-th qubit for $j > i$), we can repeat this procedure for the rest of the qubits. Hence, the circuit (5.6) indeed produces the states (5.7). However, compared to the definition of the QFT, 4, the phases of the qubits are flipped. We therefore have to apply SWAP-gates between qubit 1 and n, 2 and $n-1$, and so on.

This concludes the derivation/proof of the quantum circuit for QFT.

5.5.3 Remarks

For implementing the QFT we needed n Hadamard gates, $(n-1) + \dots + 2 + 1 = n(n-1)/2$ controlled-R_k-gates and $\lfloor n/2 \rfloor$ SWAP-gates. As a swap gates needs 3 CNOT-gates and an arbitrary controlled gates needs 2 CNOT-gates and 3 single-qubit gates, we need a total of

$$n + \frac{n(n-1)}{2} \cdot 5 + \left\lfloor \frac{n}{2} \right\rfloor \cdot 3 \in \mathcal{O}(n^2)$$

gates to implement the quantum Fourier transform. Compared to the classical discrete Fourier transform which has complexity $\mathcal{O}(n2^n)$, this is an exponential speedup! However, as usual, we cannot read out the result and need to find clever ways of applying the QFT to use its power.

The algorithm is summarized in algorithm 4.

5.6 Shor's Algorithm

In this section we will finally explore the most well-known quantum algorithm, *Shor's algorithm.* It solves the problem of finding the factors of a large number N *efficiently* and polynomial in N. The difficulty of factoring is crucial for modern-day cryptography as RSA (a very common public-key crypto-system) solely relies on the fact that factoring is *hard.* As factoring is closely related to finding the period of a function, we split this section into two parts: the first is concerned with finding the period of a special function (similar to Simon's algorithm) and the second takes the steps from period finding to factoring which is purely classical.

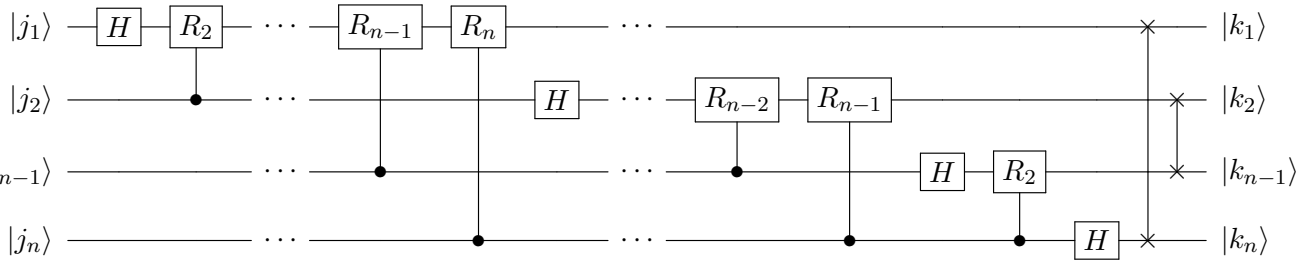
Algorithm 4: Quantum Fourier Transform

Input: state $|j_1 j_2 \dots j_{n-1} j_n\rangle$

Output: its Fourier-transformed state

// Run quantum circuit.

1



2 **return** $|k_1 k_2 \dots k_{n-1} k_n\rangle$

5.6.1 Period Finding

The first step in Shor's algorithm and the most ingenious ideas are in finding the period r of some function f under *ordinary addition* (cf. Simon's algorithm which considered modulo-two addition). The best classical algorithms solving this problem scale exponentially with the number of bits of r. With QC, we can find an algorithm that scales slightly faster than cubic.

We want to find the period r of the function $f : \{0,1\}^n \to \{0,1\}^{n_0} : x \mapsto b^x \bmod N$ such that $f(x) = f(x+r)$. We denote the number of bits of N as n_0 and let $n \coloneqq n_0$. We also have $2^{n_0} > N > r$.

Quantum Circuit

Like in Simon's algorithm, we prepare the state $|0\rangle^{\otimes n} |0\rangle^{\otimes n_0}$, apply $H^{\otimes n}$ to the first n gates, apply $U_f : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$, and measure the last n_0 qubits. This collapses our state as follows:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle \quad \xrightarrow{\text{measure}} \quad |\psi_n\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle .$$

We now have a superposition of inputs to f that all yield the same value when applying f where $0 \leq x_0 < r$ and $m \approx \lfloor 2^n / r \rfloor$. Like with Simon's algorithm, we now have the problem that a measurement destroys the superposition and we are not guaranteed to end up in the same one again and we cannot clone the state due to the no-cloning theorem. As always, we therefore need to find a clever way of learning r. The main problem is the "base" x_0 which is the changing variable (as r is constant by assumption). We only want to extract the frequency of the values—which is exactly what QFT does.

Using Quantum Fourier Transform

We saw in the previous section that we end up with the post-measurement state $|\psi_n\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle$ of which we want to extract the frequency r (i.e., we want to put it into a phase such that we are able to read

it out). We now apply QFT according to 4:

$$U_{\mathrm{FT}}\left|\psi_n\right\rangle = \frac{1}{\sqrt{m2^n}}\sum_{k=0}^{m-1} U_{\mathrm{FT}}\left|x_0+kr\right\rangle = \frac{1}{\sqrt{m2^n}}\sum_{k=0}^{m-1}\sum_{y=0}^{2^n-1} e^{2\pi i(x_0+kr)y/2^n}\left|y\right\rangle$$
$$= \frac{1}{\sqrt{m2^n}}\sum_{y=0}^{2^n-1}\sum_{k=0}^{m-1} e^{2\pi i(x_0+kr)y/2^n}\left|y\right\rangle = \sum_{y=0}^{2^n-1}\underbrace{e^{2\pi i x_0 y/2^n}}_{(\dagger)}\left[\frac{1}{\sqrt{m2^n}}\sum_{k=0}^{m-1} e^{2\pi i k r y/2^n}\right]\left|y\right\rangle$$

As x_0 only appears in the relative phase $(\dagger)$ which has no effect on the measurement (as its magnitude is 1 and the states $\left|y\right\rangle$ are orthonormal), we have successfully eliminated x_0 from our calculations! The probability of some measurement y_0 is therefore

$$P(y_0) = \frac{1}{m2^n}\left|\sum_{k=0}^{m-1} e^{2\pi i k r y/2^n}\right|^2. \tag{5.8}$$

This concludes the intermediate goal of eliminating x_0 from our mathematical framework. Next, we have to interpret our measurement outcomes by analyzing the probability (5.8) deeper.

Post-Processing

We are now at the point where we start the classical post-processing which is where the ingenious ideas that make Shor's algorithm work come into play. The result from the above quantum computations is that we measure y with probability $P(y) = \frac{1}{m2^n}\left|\sum_{k=0}^{m-1} e^{2\pi i k r y/2^n}\right|^2$. Note that we dropped the index $\cdot_0$ for brevity.

Calculating the Probability of "Success" First, we show that there is a reasonable probability to get a y "close" to an integer multiple of $2^n/r$. We say that y is close to $j2^n/r$ (the j-th multiple) if there exists an $|\delta_j| \leq 1/2$ such that

$$y = j\frac{2^n}{r} + \delta_j =: y_j.$$

We do this by inserting y_j into (5.8) and lower-bounding the probability:

$$P(y_j) = \frac{1}{m2^n}\left|\sum_{k=0}^{m-1} e^{2\pi i k r y_j/2^n}\right|^2 = \frac{1}{m2^n}\left|\sum_{k=0}^{m-1} e^{2\pi i k r(j2^n/r+\delta_j)/2^n}\right|^2 = \frac{1}{m2^n}\left|\sum_{k=0}^{m-1}\underbrace{e^{2\pi i k j}}_{=(e^{2\pi i})^{jk}=1} e^{2\pi i k r\delta_j/2^n}\right|^2$$
$$= \frac{1}{m2^n}\left|\sum_{k=0}^{m-1} e^{2\pi i k r\delta_j/2^n}\right|^2 = \frac{1}{m2^n}\left|\sum_{k=0}^{m-1}\left(e^{2\pi i r\delta_j/2^n}\right)^k\right|^2 \overset{(\dagger)}{=} \frac{1}{m2^n}\left|\frac{1-e^{2\pi i m r\delta_j/2^n}}{1-e^{2\pi i r\delta_j/2^n}}\right|^2$$
$$\overset{(\ddagger)}{=} \frac{1}{m2^n}\left|\frac{1-\cos(2\pi i m r\delta_j/2^n)-i\sin(2\pi i m r\delta_j/2^n)}{1-\cos(2\pi i r\delta_j/2^n)-i\sin(2\pi i r\delta_j/2^n)}\right|^2 \overset{(*)}{=} \frac{1}{m2^n}\frac{\sin^2(\pi m r\delta_j/2^n)}{\sin^2(\pi r\delta_j/2^n)}$$

In the marked steps, we used the following: $(\dagger)$ Geometric series, $(\ddagger)$ Euler's formula, $(*)$ the following relation:

$$\left|1-\cos(2\varphi)-i\sin(2\varphi)\right|^2 = \left(1-\cos(2\varphi)\right)^2+\sin^2(2\varphi) = 1-2\cos(2\varphi)+\cos^2(2\varphi)+\sin^2(2\varphi)$$
$$= 1-2\cos(2\varphi)+1 = 2\big(1-\cos(2\varphi)\big) = 4\sin^2(\varphi)$$

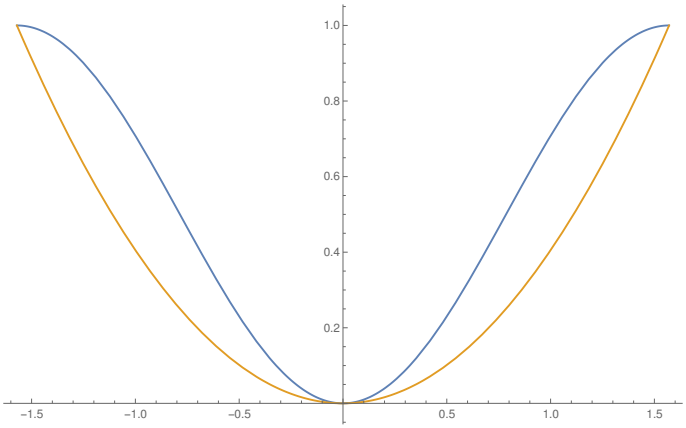


Figure 5.1: Lower bound of $\sin x$ in the interval $x \in [-\pi/2, \pi/2]$ by $x/(\pi/2)$.

With the two approximations $m \approx 2^n/r$ and $\sin x \approx x$ for small x, we arrive at

$$\frac{1}{m2^n} \frac{\sin^2(\pi i m r \delta_j / 2^n)}{\sin^2(\pi i r \delta_j / 2^n)} \approx \frac{r}{(2^n)^2} \frac{\sin^2(\pi \delta_j)}{(\pi r \delta_j / 2^n)^2} = \frac{1}{r} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j} \right)^2. \tag{5.9}$$

Let $x \coloneqq \pi \delta_j$. As we can lower-bound $\sin x$ by $x/(\pi/2)$ in the interval $x \in [-\pi/2, \pi/2]$, see Figure 5.1, we can lower-bound (5.9) as follows:

$$\frac{1}{r} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j} \right)^2 \geq \frac{1}{r} \left(\frac{2\pi \delta_j}{\pi^2 \delta_j} \right)^2 = \frac{1}{r} \left(\frac{2}{\pi} \right)^2 = \frac{2}{\pi^2 r} \approx \frac{0.405}{r}$$

As there are $r - 1$ different[4] j and the above probability is for a specific j, out final probability of getting a measurement in vicinity of an integer multiple of $2^n/r$ is greater than $40\,\%$! What remains is to recover the period from collecting multiple samples (cf. Simon's algorithm).

Recovering the Period We saw that after running the quantum circuit and measuring, we end up with ℓ samples $y_1, y_2, \ldots, y_\ell$. Suppose that these are exact, i.e., $\delta_j = 0$ for all j. We could then recover the period by calculating

$$\gcd\left(j_1 \frac{2^n}{r}, j_2 \frac{2^n}{r}, \ldots, j_\ell \frac{2^n}{r} \right) = \frac{2^n}{r}. \tag{5.10}$$

However, we do not have the exact values and the y's are just in a vicinity around integer multiples of $2^n/r$! By reformulating the "closeness" of y_j as follows and choosing a relaxed upper bound,

$$y_j = j \frac{2^n}{r} + \delta_j \quad \Longleftrightarrow \quad \left| y_j - 2^n \frac{j}{r} \right| \leq \frac{1}{2} \quad \Longrightarrow \quad \left| \frac{y_j}{2^n} - \frac{j}{r} \right| = \left| \frac{j}{r} - \frac{y_j}{2^n} \right| \leq \frac{1}{2 \cdot 2^n} = \frac{1}{2 \cdot (2^{n_0})^2} \leq \frac{1}{2r^2}, \tag{5.11}$$

we can apply the following theorem from number theory using continued fraction expansion:

Theorem 16. *If $|p/q - x| \leq 1/(2q^2)$ with real x and p/q, then p/q will appear in the CFE of x as a partial sum.*

[4] As $y_j \in \{1, 2, \ldots, 2^n - 1\}$ and also $y_j = j2^n/r + \delta_j$, we can see that j has to be one of $\{1, 2, \ldots, r-1\}$. For $j = r$, we have $y_r = r2^n/r + \delta_r = 2^n + \delta_r \geq 2^n - 1/2 > 2^n - 1$ as $|\delta_j| \leq 1/2$. For $j = 0$, we have $y_0 = 0 \cdot 2^n/r + \delta_0 = \delta_0$ and therefore $y_0 = 0$ as y_0 can only take integral values. However, we do not learn anything about r with $y = 0$ and we therefore discard it from the probability discussion. Hence, we end up with $r - 1$ "useful" values of j.

Proof. See Nielsen and Chuang, Appendix A4.4. □

With this theorem, we can recover the exact multiples $j\ 2^n/r$ with $p \triangleq j$, $q \triangleq r$, and $x \triangleq y_j/2^n$. Subsequently, we apply (5.10). We have now successfully recovered the period r with overwhelming probability! By relaxing the upper bound on the error in (5.11), the y's are not required to be *closest* to integral multiples, but they can also be the second, third, fourth, etc. closest. This upper bound ramps up the probability of measuring a useful y to $95\,\%$!

At this point we finished the period finding which was the hard part of Shor's algorithm. Next, we will look at how to go from period finding to factoring and will finally summarize the algorithm. The real clever part of Shor's algorithm lies in using (5.11) to recover the exact integral multiples.

5.6.2 From Period Finding to Factoring

After the first step (period finding), we found an r such that $f(x) = f(x+r)$ for $f(x) = b^x \bmod N$. We now use this result for finding the factors of N. We do this recursively. First, find a factor p of N. Than we know $N = pq$. Now, find the factors of p and q until all of them are prime. As N has at most $\log_2 N$ factors and finding a factor has polynomial complexity, we can solve the whole problem in polynomial time. We focus on finding *large* factors of a single number N as we can easily identify small factors such as $2, 3, 5, \ldots$ and get rid of them. With the following procedure (where all congruencies are modulo N),

1. Pick a random a. If a is not co-prime[5] to N, we found a factor and terminate.
2. Find the smallest r such that $a^{x+r} \equiv a^x$. From this we know that[6] $a^r \equiv 1$.
3. If r is odd, abort and go to first step. Otherwise, let $x \coloneqq a^{r/2} \bmod N$. From this we know that[7] $x^2 - 1 = (x-1)(x+1) \equiv 0$.
4. We know that[8] $x - 1 \not\equiv 0$, contradicting the assumption that r is the smallest period (as $r/2$ would be one, too).
5. If $x + 1 \not\equiv 0$, abort and go to the first step.
6. We now know that $p \coloneqq \gcd(x-1, N) > 1$ and $q \coloneqq \gcd(x+1, N) > 1$ due to $(x-1)(x+1) \equiv 0$. If p and q are prime, we also have $N = pq$. If they are not, we can still find factors by division.

we managed to factor N efficiently! However, at two steps (3 and 5), we have to be lucky that our assumptions hold. With some number theory, one can show that the probability of this is greater than $50\,\%$ which is not bad (see Mermin, 2007, Appendix M for details).

5.6.3 Remarks

We saw that Shor's algorithm provides an exponential speedup over the classical solution. And compared to the algorithms before, we can apply this problem to break RSA and public-key crypto! However, as we will see in section 6.6, modern-day quantum computer are far from being able to apply this algorithm to real-world problems as we have far too few qubits available (around a thousand while we need thousands of millions).

The algorithm is summarized in algorithm 5 using the period finding in algorithm 6, the actual quantum part.

[5] *Co-prime* means that $\gcd(a, N) = 1$ which is easy to check using Euclid's algorithm.
[6] Due to $a^{x+r} \equiv a^x \iff a^x a^r \equiv a^x \iff a^r \equiv 1$.
[7] Due to $a^r \equiv 1 \iff a^r - 1 \equiv 0 \iff x^2 - 1 \equiv 0$.
[8] Due to $x - 1 \equiv 0 \iff x \equiv 1 \iff a^{r/2} \equiv 1$.

Algorithm 5: Shor's Algorithm

Input: composite number N
Output: non-trivial factor of N

get rid of small factors
sample $a \in (0, N)$.
if $\gcd(a, N) > 1$ **then**
    // The GCD already is a factor of N.
    return $\gcd(a, N)$
find minimum r such that $a^r \equiv 1 \mod N$ using algorithm 6 // Quantum Part.
if r *is odd* **then**
    abort and sample new a
if $a^{r/2} + 1 \equiv 0 \mod N$ **then**
    abort and sample new a
$p, q \leftarrow \gcd(a^{r/2} \pm 1, N)$
// Return only non-trivial factors.
if $p > 1$ **then**
    return p
if $q > 1$ **then**
    return q
return "FAILED"

5.7 Grover's Algorithm

We will now look at *Grover's algorithm,* a very *different* quantum algorithm implementing unstructured search. Opposed to Shor's algorithm, however, it is much simpler and does not involve and post-processing. As before, we first go over the problem description, then the classical approach, followed by the quantum, and finally we finish up with some remarks.

5.7.1 Problem

The problem solved by Grover's algorithm is very general: suppose a list (or puzzle or whatever) with $N = 2^n$ possibilities. There are $K < N$ correct options and we want to find one of them. Let $f : \{0,1\}^n \to \{0,1\} : x \mapsto f(x)$ be a function that is 1 iff x was a solution.

5.7.2 Classical Approach

Classically, we need $N - K$ queries of f to get a solution with certainty (if we queries the first $N - K$ options, the last K options must be solutions, hence $N - K$).

Algorithm 6: Period Finding Algorithm

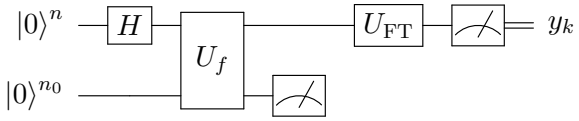
Input: function $f : \{0,1\}^n \to \{0,1\}^{n_0} = a^x \bmod N$ with the property $f(x) = f(x+r)$ for a constant period $r \in \{0,1\}^n$ with $n = 2n_0$; tries m

Output: the period r

$\mathcal{R} \leftarrow \{\}$ // Prepare period candidate set.

foreach $k = 1, 2, \ldots, m$ **do**

  // Run quantum circuit.

  if $y_k = 0$ **then**

    // A zero y does not convey any information.

    continue

  // Recover period candidates.

  $p_{k,i}/q_{k,i} \leftarrow$ irreducible numerator/denominator of i-th partial sum of $y_k/2^n$

  foreach $i = 0, 1, \ldots$ **do**

    if $q_{k,i+1} > N$ **then**

      // Add period candidate to the candiate set.

      $r_k \leftarrow q_{k,i}$

      $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_k\}$

$r^* \leftarrow \mathrm{lcm}(\mathcal{R})$ // least common multiple

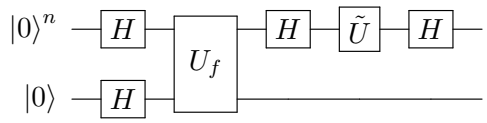
return r^*

5.7.3 Quantum Approach

We can again directly apply Deutsch's approach (5.2). As usual, we will neglect the final qubit and write the state (before the final Hadamard-layer!) as

$$\frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle .$$

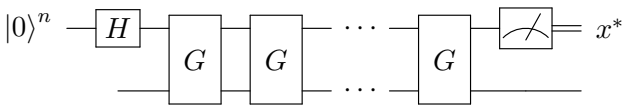
Note that the only $|x\rangle$ with a bit flip are the solutions x^* we are looking for. However, as usual, we cannot simply measure the state but have to find clever ways of reading out the information. Instead, we introduce a new unitary transformation $\tilde{U}$ followed by Hadamard layers:



This unitary transformation is defined as

$$\tilde{U} \coloneqq 2 |0\rangle^n {}^n\langle 0| - \mathbb{1} = \mathrm{diag}(1, -1, -1, \ldots, -1).$$

For brevity, we write $U_R \coloneqq H^{\otimes n} \tilde{U} H^{\otimes n}$ and $G \coloneqq U_R U_f$ which we call *Grover's unitary*. We claim (and show) now that applying G for $\sqrt{N}$ times,



yields a correct result $f(x^*) = 1$ when measuring with almost unit probability. We will now first look an at algebraic derivation why this is the case and then look at a graphical illustration.

Algebraic Derivation

First, we derive what U_R actually does. Let $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$ be the equal superposition of all basis states (note that this is also the input to the first Grover-unitary). Then we can write U_R as

$$H^{\otimes n} \tilde{U} H^{\otimes n} = H^{\otimes n} \big(2 |0\rangle^n {}^n\langle 0| - \mathbb{1}\big) H^{\otimes n} = 2\big(H^{\otimes n} |0\rangle^n\big) \big({}^n\langle 0| H^{\otimes n}\big) - \mathbb{1} = 2 |\psi\rangle\langle\psi| - \mathbb{1}.$$

For further derivations, let

$$|\alpha\rangle \coloneqq \frac{1}{\sqrt{N-K}} \sum_{x,\, f(x)=0} |x\rangle \qquad\qquad |\beta\rangle \coloneqq \frac{1}{\sqrt{K}} \sum_{x,\, f(x)=1} |x\rangle$$

be states in superposition of all the non-solutions and solutions, respectively. We can then reformulate $|\psi\rangle$ as

$$|\psi\rangle = \sqrt{\frac{N-K}{N}} |\alpha\rangle + \sqrt{\frac{K}{N}} |\beta\rangle \doteq \cos\frac{\theta}{2} |\alpha\rangle + \sin\frac{\theta}{2} |\beta\rangle$$

with an appropriate θ. By applying $G = U_r U_f$, we have

$$\begin{aligned}
U_R U_f |\psi\rangle &= U_R\left[\cos\frac{\theta}{2}|\alpha\rangle - \sin\frac{\theta}{2}|\beta\rangle\right] = U_R\left[\cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle - 2\sin\frac{\theta}{2}|\beta\rangle\right] = U_R\left[|\psi\rangle - 2\sin\frac{\theta}{2}|\beta\rangle\right] \\
&= \left(2|\psi\rangle\langle\psi| - \mathbb{1}\right)\left[|\psi\rangle - 2\sin\frac{\theta}{2}|\beta\rangle\right] = 2|\psi\rangle\underbrace{\langle\psi|\psi\rangle}_{=1} - 4\sin\frac{\theta}{2}|\psi\rangle\underbrace{\langle\psi|\beta\rangle}_{=\sin(\theta/2)} - |\psi\rangle + 2\sin\frac{\theta}{2}|\beta\rangle \\
&= 2|\psi\rangle - 4\sin^2\frac{\theta}{2}|\psi\rangle - |\psi\rangle + 2\sin\frac{\theta}{2}|\beta\rangle = |\psi\rangle - 4\sin^2\frac{\theta}{2}|\psi\rangle + 2\sin\frac{\theta}{2}|\beta\rangle \\
&= \cos\frac{\theta}{2}|\alpha\rangle + 3\sin\frac{\theta}{2}|\beta\rangle - 4\sin^2\frac{\theta}{2}\left(\cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle\right) \\
&= \cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle - 4\sin^2\frac{\theta}{2}\cos\frac{\theta}{2}|\alpha\rangle - 4\sin^2\frac{\theta}{2}\sin\frac{\theta}{2}|\beta\rangle + 2\sin\frac{\theta}{2}|\beta\rangle \\
&= \left(1 - 4\sin^2\frac{\theta}{2}\right)\cos\frac{\theta}{2}|\alpha\rangle + \left(3 - 4\sin^2\frac{\theta}{2}\right)\sin\frac{\theta}{2}|\beta\rangle = \cos\frac{3\theta}{2}|\alpha\rangle + \sin\frac{3\theta}{2}|\beta\rangle
\end{aligned}$$

where we uses some trigonometric identities in the last step. Furthermore, we can show (by induction) that after k steps, we have the state

$$G^k|\psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right)|\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right)|\beta\rangle\,.$$

That is, with every k, we rotate around the non-solutions $|\alpha\rangle$ and the solutions $|\beta\rangle$. So how many iterations k do we need to get a success probability close to one? If we approximate $|\beta\rangle$'s amplitude with one, we can derive what k should be:

$$\sin\left(\frac{2k+1}{2}\theta\right) \approx 1 \quad \Longrightarrow \quad \frac{2k+1}{2}\theta \approx \frac{\pi}{2} \quad \Longrightarrow \quad k \approx \frac{\pi}{2\theta} - \frac{1}{2} \approx \frac{\pi}{2\theta} \tag{5.12}$$

Note that the last approximation is valid as k is an integer anyway and we can always use the ceiling. If we assume that the initial probability is small, i.e., θ is small, we can approximate it:

$$\sin\frac{\theta}{2} \approx \frac{\theta}{2} \overset{!}{=} \sqrt{\frac{K}{N}} \quad \Longrightarrow \quad \theta \approx 2\sqrt{\frac{K}{N}}$$

Plugging this result back into (5.12) yields

$$k \approx \frac{\pi}{2\theta} \approx \frac{\pi}{2\cdot 2\sqrt{\frac{K}{N}}} = \frac{\pi}{4}\sqrt{\frac{N}{K}}$$

Grover's algorithm is therefore in $\mathcal{O}(\sqrt{N/K})$. As θ gets scales with K, fewer iterations are necessary to find a solution in a pool of multiple solutions. Also, the probability oscillates with a higher frequency. Interestingly, applying G too often reduces the probability of getting a correct readout again! But on a large scale, this is not a problem as we can always check a solution by invoking f for the measurement outcome.

Illustration

5.7.4 Remarks

We saw that Grover's algorithm provides a square-root speedup over the classical solution and an even greater speedup if there are multiple solutions. Compared to Shor's algorithm, no post-processing is necessary and

not so many qubits are necessary. Also, Grover's algorithm is a general-purpose search algorithm as long as we can implement the query unitary. However, to determine the number of G-applications, we must know how many solutions exist. If we do not know this, we have to guess and possibly run the algorithm over and over again. We also have the following theorem:

Theorem 17 (BBBV Theorem)**.** *Grover's algorithm is optimal for black-box unstructured search problems.*

If there would exist a more efficient algorithm, we could solve all problems in NP.
The algorithm is summarized in algorithm 7.

Algorithm 7: Grover's Algorithm

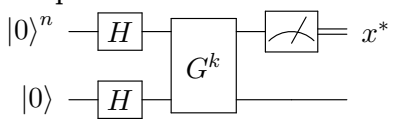
Input: function $f : \{0,1\}^n \to \{0,1\} : x \mapsto f(x)$ which is 1 for a solution; optional: number of solutions K or number of iterations k

Output: a solution x^* with almost unit probability if K is given

1 **if** *K is given* **then**

2 $\quad k = \lceil \frac{\pi}{4} \cdot \sqrt{\frac{N}{K}} \rceil$ // Calculate number of iterations.

// Run quantum circuit.

3 

4 **return** x^*

5.7.5 Modified Grover's Algorithm

6 Quantum Error Correction

So far, we assumed that applying gates works perfectly, i.e., that the state transform exactly as our mathematical model predicts. However, in the real world, (quantum) computers are noisy and by no means perfect. In classical computing, the errors that might occur when sending a message (e.g., through the internet or through time by saving it to disk) are confined to bit-flips. In order to make a computer resilient towards bit-flips, the most simplistic approach is to just copy the data and define *logical bits*

$$0 \mapsto 000 =: 0_L \qquad\qquad 1 \mapsto 111 =: 1_L.$$

This is called *encoding* and the protocol is called the *code* (in this case, *repetition* code). Decoding the message works by a majority vote, i.e.:

$$\begin{aligned} 000 &\mapsto 0 & 001 &\mapsto 0 & 010 &\mapsto 0 & 011 &\mapsto 1 \\ 100 &\mapsto 0 & 101 &\mapsto 1 & 110 &\mapsto 1 & 111 &\mapsto 1 \end{aligned}$$

This means that a decoding error happens only if more than one bit is flipped. In other words: the encoding is resilient w.r.t. to one bit-flip. Assume that a bit-flip happens with probability p (see Figure 6.1a). Then the error probability is

$$p_e^{(1)} \coloneqq P(\text{two or more flipped}) = 3P(\text{exactly two flipped}) + P(\text{all flipped}) = 3p^2(1-p) + p^3 < 3p^2,$$

where $3p^2$ is an upper bound. If we re-apply the encoding (and therefore encode a logical bit into nine physical bits), the new error probability is $p_e^{(2)} = 3(3p^2)^2$. In general, after k applications of the encoding, the error probability is proportional to $(3p)^{2^k}$. Hence, $(3p)^{2^k} \to_{k\to\infty} 0$ for $p < 1/3$ and we can achieve an arbitrarily good encoding for finite k.

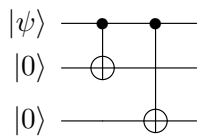
But can we directly transfer this principle to quantum error correction (QEC)? No, because:

- We cannot copy data due to the no-cloning theorem—but would it even help as measurements destroy everything anyway?
- We do not have discrete error but a continuum—do we need infinite precision to locate the error?
- Measuring destroys the message—how to do decode?

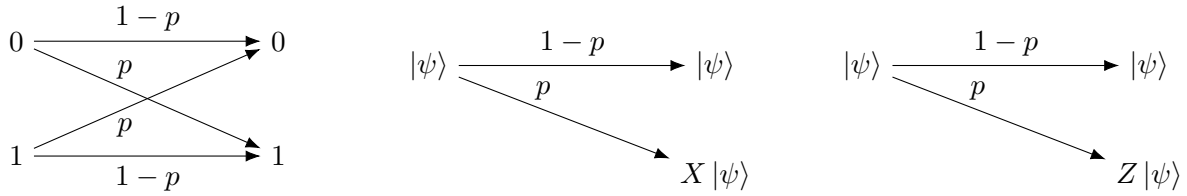
Fortunately though, all of these problems can be circumvented and QEC is possible. We will now cover two kinds of errors: bit-flips and phase-flips and will subsequently see that correcting these suffices to correct all possible errors, effectively reducing the continuum of errors to a finite set.

6.1 Tackling Bit-Flips

We start by fixing bit-flips. To encode the message, we copy the coefficients of an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to two encoding qubits:



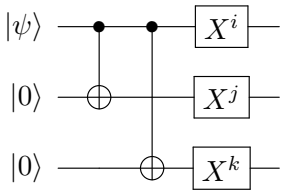
$$= \alpha|000\rangle + \beta|000\rangle =: \alpha|0\rangle_L + \beta|1\rangle_L$$



(a) Classical Bit-Flip (b) Quantum Phase-Flip (c) Quantum Bit-Flip

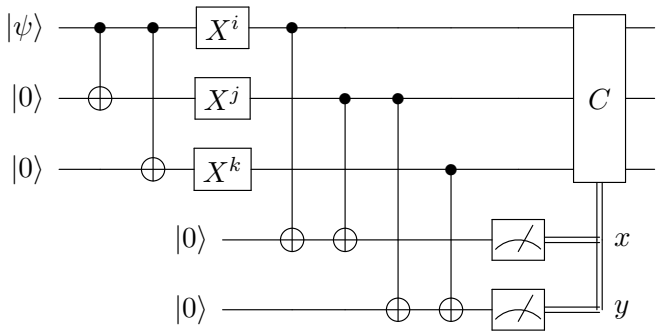
Figure 6.1: State diagram for classical/quantum bit/phase-flip channels while sending a message. A bit/phase-flip happens with probability p and the message is left untouched with probability $1-p$.

Note that we did not actually copy the state, but only the *coefficients* which does not violate the no-cloning theorem! We now assume that an error only happens on exactly one qubit (because otherwise we are not able to recover anyway). We encode this as follows:



i	j	k	State
0	0	0	$\alpha\|000\rangle + \beta\|111\rangle$
1	0	0	$\alpha\|100\rangle + \beta\|011\rangle$
0	1	0	$\alpha\|010\rangle + \beta\|101\rangle$
0	0	1	$\alpha\|001\rangle + \beta\|110\rangle$

Fortunately, all possible result states are orthogonal! Hence, we can uniquely identify my measuring even though they are all in superposition. However, if we would simply measure, we would destroy the state. We therefore copy the state again into two *helper qubits* which we can then measure:



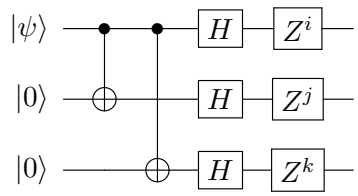
x	y	Correction C
0	0	$\mathbb{1}$
0	1	X_3
1	0	X_1
1	1	X_2

The interpretation of this is as follows: for $x = 1$, we conclude that the first and second qubit have different values. Similarly, $y = 1$ encodes that the second and third qubit have different values. If they have the same value, the CNOT-gates would either be not applied or cancel each other out. Note again that the sole reason why this works is that the state are orthogonal! From this discussion, we can deduce the above corrections.

This approach is also called a *parity check* of Z_1Z_2 and Z_2Z_3, i.e., a parity check in Z-basis.

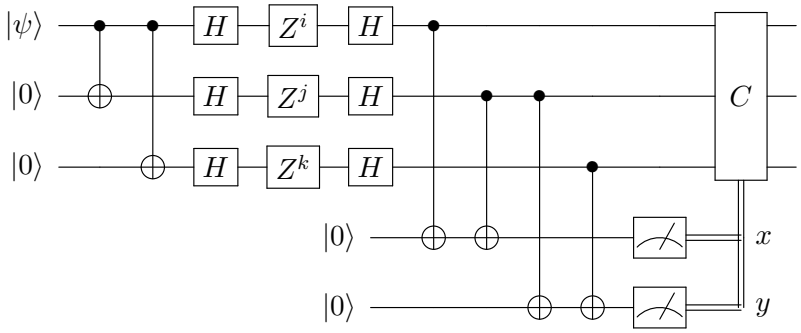
6.2 Tackling Phase-Flips

Correcting phase-flips (see Figure 6.1c) is analogous to correcting bit-flips, but in Hadamard basis. The encoding circuit including the error model along with the resulting states therefore is:



i	j	k	State
0	0	0	$\alpha\|+++\rangle + \beta\|---\rangle$
1	0	0	$\alpha\|-++\rangle + \beta\|+--\rangle$
0	1	0	$\alpha\|+-+\rangle + \beta\|-+-\rangle$
0	0	1	$\alpha\|++-\rangle + \beta\|--+\rangle$

By applying Hadamard again in the decoding circuit, the identity $HZH = X$ directly tells us that we need to apply the same corrections as for bit-flips:



x	y	Correction C
0	0	$\mathbb{1}$
0	1	X_3
1	0	X_1
1	1	X_2

This approach is a parity check of X_1X_2 and X_2X_3, i.e., a parity check in X-basis.

6.3 Shor's Code

The idea of *Shor's code* is to concatenate the circuits discussed before, using the bit-flip correction three times for each physical qubit of the phase-flip parity check. The whole circuit is shown in Figure 6.2. One can easily verify that Shor's code used the following mapping for logical qubits:

$$\begin{aligned} |0\rangle &\mapsto \frac{|000\rangle + |111\rangle}{\sqrt{2}} \otimes \frac{|000\rangle + |111\rangle}{\sqrt{2}} \otimes \frac{|000\rangle + |111\rangle}{\sqrt{2}} =: |0\rangle_L \\ |1\rangle &\mapsto \frac{|000\rangle - |111\rangle}{\sqrt{2}} \otimes \frac{|000\rangle - |111\rangle}{\sqrt{2}} \otimes \frac{|000\rangle - |111\rangle}{\sqrt{2}} =: |1\rangle_L \end{aligned} \tag{6.1}$$

An important feature of Shor's code is that it is not only capable of correcting X or Z errors, but also XZ errors, i.e., simultaneous bit- and phase-flips. With this feature, it is possible to correct any possible error (see subsection 6.3.1). However, Shor's code is not optimal as it uses so many qubits. It is also possible to use only seven or even five qubits, see section 6.4.

6.3.1 Universal Error Correction

As already discussed is the Shor code capable of correcting XZ-type errors acting on a single qubit. From this we can conclude that the Shor code is in fact capable of correcting arbitrary errors, even non-unitary ones, as long as they act on a single qubit and are valid quantum operations.

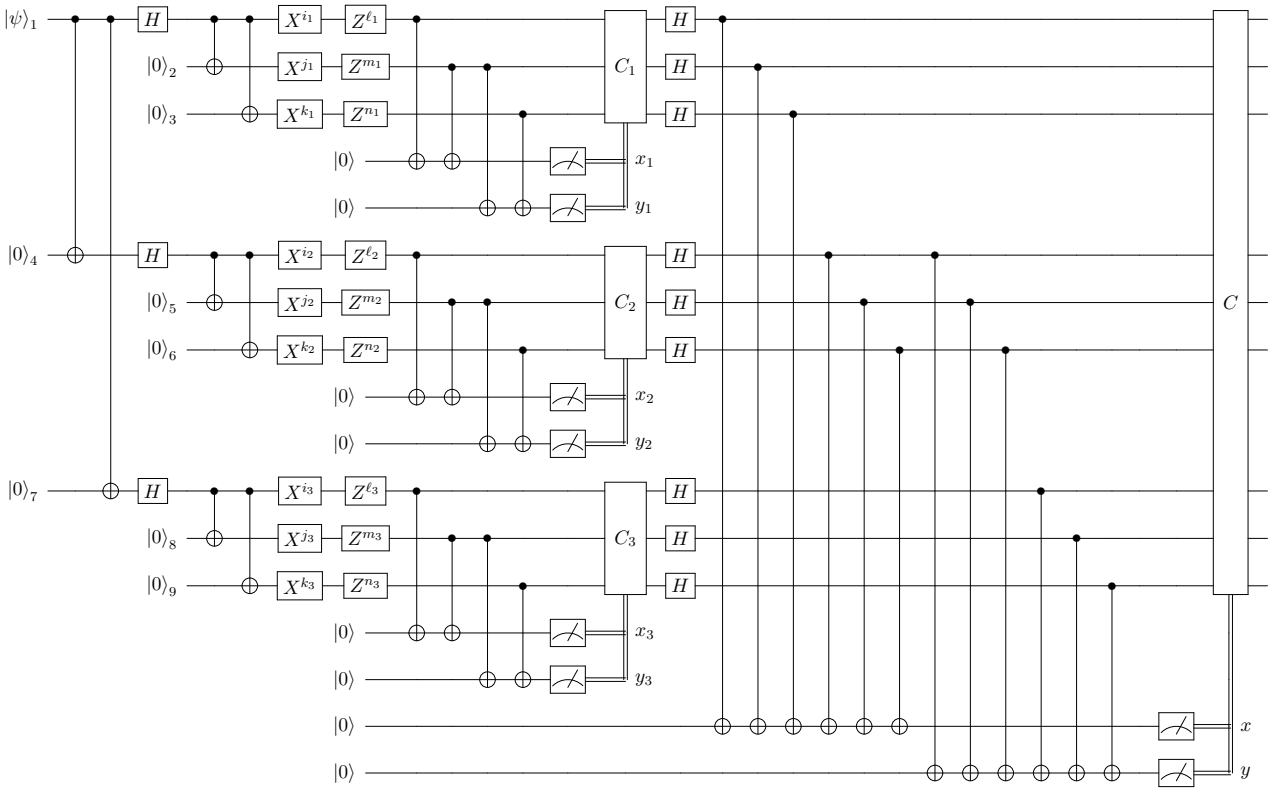


Figure 6.2: Quantum circuit for encoding in Shor's code (the left part until the error gates X and Z) and for decoding (everything to the right of the error gates). The matrices C_1, C_2, C_3, and C denote the corresponding correction matrices for the bit flip ("inner" code) and the phase flip ("outer" code). The "error indices" are mutually exclusive per index or all zero, i.e., $(i_a, j_a, k_a), (\ell_b, m_b, n_b) \in \{(0,0,0), (1,0,0), (0,1,0), (0,0,1)\}$ for $a, b \in \{1,2,3\}$.

Theorem 18 (Universal Error Correct)**.** *The Shor code is capable of correcting arbitrary errors.*

Proof. W.l.o.g, assume that the error happens on the first qubit and is described by $E_1 \in \mathbb{C}^{2\times 2}$. We can express this matrix using the three Pauli matrices:

$$E = \alpha_0 \mathbb{1}_1 + \alpha_1 X_1 + \alpha_2 Z_1 + \alpha_3 X_1 Z_1$$

This is easy to verify by calculating the right-hand-side, equating it to E_1, and explicitly solving the resulting linear system of equations:

$$\begin{bmatrix} e_{11} & e_{21} \\ e_{21} & e_{22} \end{bmatrix} = \begin{bmatrix} \alpha_0 + \alpha_2 & \alpha_1 - \alpha_3 \\ \alpha_1 + \alpha_3 & \alpha_0 - \alpha_2 \end{bmatrix} \qquad \Longrightarrow \qquad \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} e_{11} + e_{22} \\ e_{12} + e_{21} \\ e_{11} - e_{22} \\ e_{21} - e_{12} \end{bmatrix}$$

Let $|\psi\rangle_L = \alpha |0\rangle_L + \beta |1\rangle_L$ be a logical qubit encoded with Shor's code (see (6.1) for the logical basis states). Then, by linearity, an application of the error operator E_1 yields

$$E_1 |\psi\rangle_L = \alpha_0 |\psi\rangle_L + \alpha_1 X_1 |\psi\rangle_L + \alpha_2 Z_1 |\psi\rangle_L + \alpha_3 X_1 Z_1 |\psi\rangle_L .$$

By dropping the last six qubits from the basis states (as the error only acts on the first qubit), each individual error has the following effect on $|\psi\rangle_K$:

$$\begin{aligned}
\mathbb{1}_1 |\psi\rangle_L &= \frac{\alpha}{\sqrt{2}}(|000\rangle + |111\rangle) \otimes |\cdots\rangle + \frac{\beta}{\sqrt{2}}(|000\rangle - |111\rangle) \otimes |\cdots\rangle \\
X_1 |\psi\rangle_L &= \frac{\alpha}{\sqrt{2}}(|100\rangle + |011\rangle) \otimes |\cdots\rangle + \frac{\beta}{\sqrt{2}}(|100\rangle - |011\rangle) \otimes |\cdots\rangle \\
Z_1 |\psi\rangle_L &= \frac{\alpha}{\sqrt{2}}(|000\rangle - |111\rangle) \otimes |\cdots\rangle + \frac{\beta}{\sqrt{2}}(|000\rangle + |111\rangle) \otimes |\cdots\rangle \\
X_1 Z_1 |\psi\rangle_L &= \frac{\alpha}{\sqrt{2}}(|100\rangle - |011\rangle) \otimes |\cdots\rangle + \frac{\beta}{\sqrt{2}}(|100\rangle + |011\rangle) \otimes |\cdots\rangle
\end{aligned}$$

As these states are mutually orthogonal, measuring the parity qubits uniquely identifies one of the four errors and the state collapses onto either one of them. Subsequently, Shor's code can be used to correct the individual errors. Hence, Shor's code can be used to correct arbitrary errors. □

6.4 Other Codes

As already mentioned, the Shor code is not *optimal* in the sense of saving qubits. There exist several smaller ones, for instance the *Steane code*

$$|0\rangle_L = \frac{1}{\sqrt{8}}(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle)$$

$$|1\rangle_L = \frac{1}{\sqrt{8}}(|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle)$$

or five-qubit codes. However, the Steane code has some nice properties which we will discuss in the next section.

6.5 Fault-Tolerance and Transversality

In fault-tolerant QC, the idea is to perform all operations in encoded states (i.e., logical qubits), without ever decoding. One code supporting this is the Steane code we saw in the previous section. It allows us to execute certain gates *transversely*. That is, we can design a gate that has the same effect on the individual physical qubits as on the logical qubits. For the Steane code, the basic gates are simply tensor-products of single-qubit gates:

$$X_L = X_1 \otimes X_2 \otimes \cdots \otimes X_7 \qquad Z_L = Z_1 \otimes Z_2 \otimes \cdots \otimes Z_7 \qquad H_L = H_1 \otimes H_2 \otimes \cdots \otimes H_7$$

While they acting on the individual qubits is trivial, one can also show that:

$$\begin{aligned} X_L |0\rangle_L &= |1\rangle_L & X_L |1\rangle_L &= |0\rangle_L \\ Z_L |0\rangle_L &= |0\rangle_L & Z_L |1\rangle_L &= -|1\rangle_L \\ H_L |0\rangle_L &= \frac{1}{\sqrt{2}}(|0\rangle_L + |1\rangle_L) = |+\rangle_L & H_L |1\rangle_L &= \frac{1}{\sqrt{2}}(|0\rangle_L - |1\rangle_K) = |-\rangle_L \end{aligned}$$

An important property to consider in fault-tolerant QC is how errors propagate. For the X-gate, for instance, an error does not propagate to other physical qubits as the gate is local and can be corrected with the Steane code. Application of a CNOT-gate withing a Steane block, however, may propagate an error onto a second qubit which can not be corrected! Instead, a logical CNOT-gate

$$CNOT_L |x\rangle_L |y\rangle_L \mapsto |x\rangle_L |x \oplus y\rangle_L = CNOT_{1,8} \otimes CNOT_{2,9} \otimes \cdots \otimes CNOT_{7,14}$$

has to be used that only works across two blocks (see Figure 6.3 for the circuit). The T-gate, however, cannot be implemented transversely. Its fault-tolerant implementation is more complicated and out of scope for this course.

The general scheme of performing fault-tolerant QC is:

1. prepare quantum state in logical qubit
2. apply fault-tolerant gate
3. correct errors
4. go to second step until the algorithm is finished

As we already saw, the second step is the fundamental challenge as it is hard to come up with fault-tolerant gates, which we define as follows:

Definition 5 (Fault-Tolerance)**.** A procedure is called *fault-tolerant* if the failure of one component causes at most one error in each encoded block at the output step.

6.6 Threshold Theorem

So far, we assumed that the QEC itself works perfectly. But what if the error correction itself causes errors again? Like for classical repetition codes, we can simply re-encode the bits, building up a layered coding architecture. Let m be the number of places where an error can occur, then the probability of an error using a single coding layer is upper-bounded by $p_e^{(1)} = (mp)^2$. For k layers, the error probability is $p_e^{(k)} = (mp)^{2^k}$ which, analogous to classical error correction, converges to zero for $k \to \infty$ if $p < 1/m$. Hence, we can achieve any error probability ϵ with a large enough k.

This brings us to the *threshold theorem:*

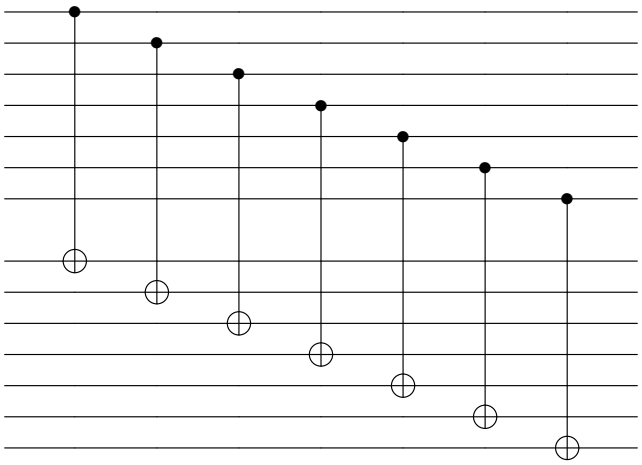


Figure 6.3: Fault-Tolerant CNOT-Gate

Theorem 19 (Threshold Theorem)**.** *A quantum circuit with $p(n)$ gates on n qubits may be simulated with an error probability of at most $\epsilon > 0$ using*[1]

$$\mathcal{O}\left(p(n)\ \log^c \frac{p(n)}{\epsilon}\right)$$

gates (for some constant c when the gates fail with probability of at most p given that $p < p_{\text{th}}$ for some code-dependent constant threshold p_{th}.

For the Steane code, $p_{\text{th}} \approx 10^{-5}$. Hence, to factor numbers with around 2000, we would needs millions of millions of qubits! This is the sole reason why we do not have quantum computing at the moment!

[1] Note that in the lecture, this bound is written as $\mathcal{O}\big(p(n)\ \text{poly}(\log(p(n)/\epsilon))\big)$.

7 Quantum Nonlocality

7.1 Elements of Reality

7.2 CHSH Inequality

7.3 Quantum Violation of the CHSH Inequality

7.4 Tsirelson's Bound and Quantum Key Distribution

8 Measurement-Based Quantum Computing

8.1 Identity

8.2 Arbitrary Rotations

8.3 CNOT

8.4 Cluster States

8.5 Handling Errors

8.6 Important Gates