Introduction to Quantum Computing Lecture Notes

Ahmet Çevik

Contents

1 Introduction

Quantum mechanics is a deeply strange theory that challenges the traditional notions about the physical reality that we know of the macroscale world. It is inherently probabilistic and it forbids us from measuring certain kinds of physical quantities. For example, due to *Heisenberg's uncertainity principle* we cannot know, for certain, both the momentum and the position of a subatomic particle. It forbids us from asking questions about the physical system in consideration. If a subatomic particle has travelled from point A to point B in a trajectory motion, we may not know which path it followed. Yet quantum mechanics has been very successful in explaning physical reality, such as the conductivity of metals, transparency of glass, colors, chemical reactions and etc.

Classical computing has served us well for many purposes in the history of science. Computations relied on the rules of Newtonian physics. The seminal work of Einstein, Schrödinger, Dirac and other influential scientists about a new kind of physics tells us that it may actually be time for a change towards a different computational paradigm.

Quantum computing is a type of natural computing which uses subatomic matter and quantum mechanical principles that we will explain shortly such as *superposition*, *interference* and *entanglement* to solve various problems more efficiently than classical algorithms. It is worth noting beforehand that quantum computers do not solve problems that classical computers fail to solve. In other words, quantum computers do not have any computational power over classical computers as they compute the same class of functions. It is just that quantum computers solve certain hard problems more efficiently than classical computers. This does not to say that classical computers cannot solve these hard problems given a sufficient amount of time.

Quantum computational models give us a considerable amount of asymptotic speed-up in solving different kinds of problems including unordered searching, integer factorization, period finding and many others. For our purpose, we will look at popular quantum algorithms starting from the simplest one, moving to more complicated quantum algorithms.

The content of this lecture notes is taken from multiple sources, but the topics are carefully chosen with the aim of teaching the most "standard" subjects in a typical quantum computing course.

We assume some basic familiarity with linear algebraic notions. We shall also assume some high school trigonometry and complex numbers. Necessary definitions will be given in this chapter. No background on quantum mechanics is assumed, although it would be certainly helpful to understand the intuition behind the course material. It is also encouraged that whenever the reader is not clear about some point, she could quickly skim through the rest of the chapter as the answer to those points might be provided at a later paragraph.

1.1 Classical Deterministic Systems

In the next two subsections we will show how matrices can be used in computing systems. The reader may skip to Section 1.3 without loss of generality. Now consider children's marbles placed on the vertices of a graph.

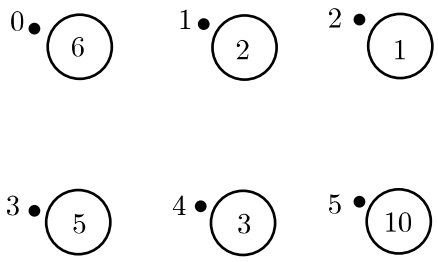


Figure 1: A deterministic state.

We shall denote the state of this system via the column vector

$$M = \begin{bmatrix} 6 \\ 2 \\ 1 \\ 5 \\ 3 \\ 10 \end{bmatrix}.$$

We may also denote it by the row matrix to save some space $M^T = [6, 2, 1, 5, 3, 10]$, where M^T is the *transpose* of M defined as $M[i, j]^T = M[j, i]$, but we will mostly use the column matrix form.

We are also interested in the dynamics of the system, i.e. how states change. Let us represent this by the following directed graph.

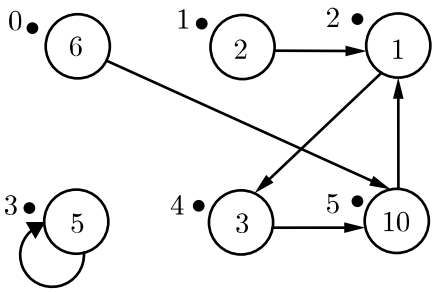


Figure 2: Dynamics of the system.

If there is an arrow from vertex i to vertex j, then let us say in one time click all the marbles on vertex i will move to vertex j. Since we are considered with deterministic systems in this example, the types of graphs we shall be concerned with are those with exactly one outgoing edge from each vertex.

The adjacency matrix of this graph is equivalent to the matrix M

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

where $M[i, j] = 1$ if and only if there is an arrow from vertex j to vertex i.

Let us multiply M by the state $X = [6, 2, 1, 5, 3, 10]^T$. Then we have,

$$MX = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 1 \\ 5 \\ 3 \\ 10 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 12 \\ 5 \\ 1 \\ 9 \end{bmatrix} = Y$$

This corresponds to the fact that if X describes the state of the system at stage s, then Y is the state at stage $s+1$. Therefore, M is a way of describing how the state of the system changes. Finite dimensional quantum systems work in the same way. We may also apply the matrix M n many times consecutively on the state to see how the system changes after n stages.

1.2 Classical Probabilistic Systems

First thing to note is that quantum mechanics is not deterministic. That is, neither the state of the system nor the dynamics of the system are deterministic. There is always an indeterminacy in our knowledge of a state. Moreover, the states change with probabilistic laws as opposed to deterministic laws.

Continuing our example, now let us deal with a single marble. The state of the system will tell us the probabilities of the single marble being on each vertex. For a 3-vertex graph, a typical state would look like $X = [\frac{1}{5}, \frac{3}{10}, \frac{1}{2}]^T$. This will correspond to the fact that there is 1/5 chance that the marble is on vertex 0, and 3/10 chance on vertex 1, and 1/2 chance on vertex 2. The marble must be somewhere in the graph. So the sum of the probabilities is 1.

We should also generalize the dynamics of the system. Instead of having exactly one edge leaving each vertex, we allow multiple edges with the condition that the sum of the probabilities of outgoing and incoming edges add up to 1.

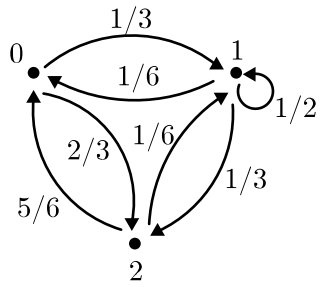


Figure 3: An example to a probabilistic system dynamics.

The matrix representation is

$$M = \begin{bmatrix} 0 & \frac{1}{6} & \frac{5}{6} \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{6} \\ \frac{2}{3} & \frac{1}{3} & 0 \end{bmatrix}.$$

Suppose that we have a state $X = [\frac{1}{6}, \frac{1}{6}, \frac{2}{3}]^T$. If we apply M on X we get $MX = Y$ which is

$$\begin{bmatrix} 0 & \frac{1}{6} & \frac{5}{6} \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{6} \\ \frac{2}{3} & \frac{1}{3} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{6} \\ \frac{1}{6} \\ \frac{2}{3} \end{bmatrix} = \begin{bmatrix} \frac{21}{36} \\ \frac{9}{36} \\ \frac{6}{36} \end{bmatrix} = Y$$

Quantum systems are generally in a probabilistic state. Manipulating the system will correspond to multiplying the state by a matrix. Each computation step will correspond to one matrix transformation. The resulting vector will describe the state of the system.

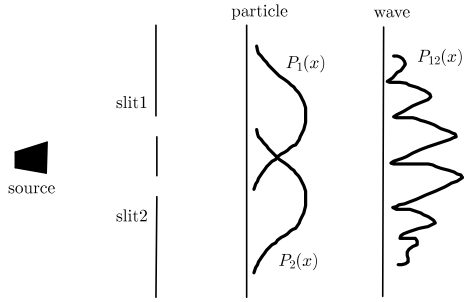
1.3 Quantum systems

In 1801, Thomas Young conducted the *double-slit experiment* that explained the concept of wave-particle duality, which later gave us hints regarding the behavior of matter at the subatomic level. The experiment also helped us to understand the physics of the microscopic world. Subatomic matters are neither particles nor waves, but they behave in their own quantum mechanical way. The double-slit experiment demonstrates that a subatomic particle of matter behaves sometimes like a macroscale particle and sometimes like

a wave. It also shows that the very act of observing a subatomic particle has dramatic effects on its behavior.

In this experiment, we have a source emitting light or electrons. Let us suppose that intensity of the source is so low that it emits the photons or electrons as discrete particles once in every second, let us say. We also have a panel with two slits, and along way from the panel there is a screen on which we measure the energy of the source admitted at some point x on the screen. What is the probability that an electron is detected at point x?

First, if we were to shoot machine gun bullets, what would happen in the macroscale? If only one slit is opened, we observe that bullet hole will more likely to be detected on the screen closer to position of the slit.



What happens if we open both slits? Naturally a similar behavior will be observed for macroscale objects. Now what about the behavior of the light? If one slit is opened, we have no problem. It behaves exactly like a bullet. When two slits are opened, it forms an interference pattern on the screen.

Now if we were to shoot photons, discrete particles of light, we could expect that it would behave like a bullet. The strange part is that it forms the exact interference pattern.

The double-slit experiment demonstrates that physics at the subatomic level is a lot different than that of the macroscale level. In the next subsection we shall we looking at how to model this physics mathematically for the purpose of our study.

1.4 Complex Numbers and Linear Algebra

Complex numbers play a major role in quantum computing. A complex number is a number in the form of $c = a + bi$, where a and b are real numbers and $i^2 = -1$. Given that, we see that $i^3 = i \cdot i^2 = -i$, and $i^4 = 1$.

In fact, $i^n = i^{n \bmod 4}$. In quantum computing, probabilities of states and transitions are given in the form of a complex number $c = a + bi$ such that $|c|^2 = a^2 + b^2$ is a real number between 0 and 1. We call $|c|$ the *modulus* of c, i.e., $|c| = \sqrt{a^2 + b^2}$.

Why complex numbers? The necessary reason has to do with time evolution of quantum systems in *Schrödinger's equation*. The auxiliary reason is that real number probabilities necessarily add together to obtain larger real numbers. On the other hand, complex numbers can *cancel* each other and lower their probability. In detail, if p_1 and p_2 are two real numbers between 0 and 1, then $(p_1 + p_2) \geq p_1$ and $(p_1 + p_2) \geq p_2$. However, if c_1 and c_2 are two complex numbers, $|c_1 + c_2|^2$ need not be bigger than $|c_1|^2$ or than $|c_2|^2$.

Example. Let $c_1 = 5 + 3i$ and let $c_2 = -3 - 2i$. Then $|c_1|^2 = 34$ and $|c_2|^2 = 13$ but $|c_1 + c_2|^2 = |2 + i|^2 = 5$. Notice that $|c_1 + c_2|^2 < |c_1|^2$.

But why do we need this kind of probability behavior? Subatomic particles both behave like a particle and a wave. When two waves interfere, they either cancel or reinforce each other (see Double Slit Experiment). The former is a *destructive interference* whereas the latter is called *constructive interference*. Complex numbers are used, for convenience, to model the *interference* phenomenon in quantum mechanics. In fact, complex numbers are necessary in quantum computing. As we will discuss later, this has to do with time evolution and reversibility.

Given a complex number $c = a + bi$, we denote the *complex conjugate* of c by $c^* = a - bi$. The set of complex numbers is denoted by $\mathbb{C}$ and the set of real numbers is denoted by $\mathbb{R}$.

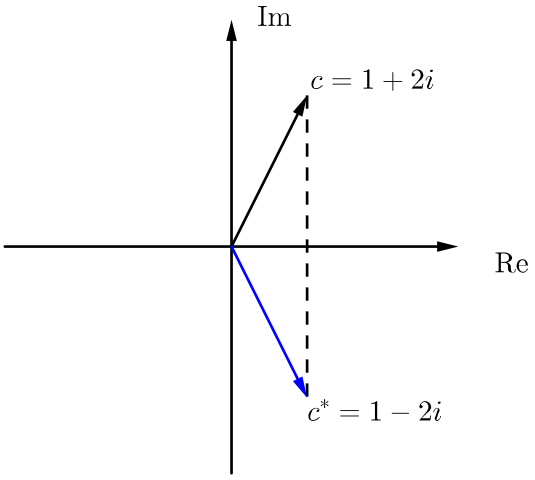


Figure 4: Complex conjugate of a complex number is just a reflecting its imaginary part around the real axis.

The reader should note that any complex number $c = a + bi$ can also be represented by polar coordinates (ρ, θ), where ρ is called the *magnitude* and θ is called the *phase*. Given $c = a + bi$ such that $a, b \in \mathbb{R}$, from elementary trigonometry we compute ρ and θ as

$$\rho = \sqrt{(a^2 + b^2)}, \qquad \theta = \arctan\left(\frac{b}{a}\right)$$

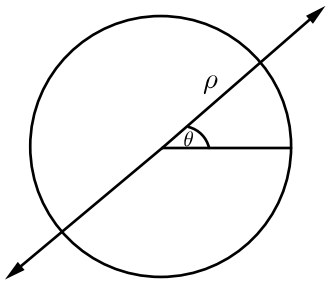


Figure 5: Polar representation of a complex number. The phase gives the degree and the magnitude gives the length of the vector.

We can go back from polar to Cartesian representation, again using trigonometry,

$$a = \rho\cos(\theta), \qquad b = \rho\sin(\theta).$$

We now give a few definitions of some linear algebraic concepts that we will use.

Definition 1. A *group* $(G, \cdot)$ is a set G together with a binary operator $\cdot$ that satisfies the following axioms:

1. For all elements $a, b, c \in G$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
2. There exists an *identity element* $e \in G$, such that for any element $a \in G$, $a \cdot e = e \cdot a = a$.
3. For every element $a \in G$, there exists an inverse element $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

If a group G also satisfies that for all elements $a \cdot b = b \cdot a$, then G is called an *abelian group*.

Definition 2. Let G be a group. If H is a group and $H \subset G$, then H is called a *subgroup* of G. If H is a subgroup of G, then for each $g \in G$, the set

$$gH = \{gh : h \in H\}$$

is called a *coset* of H (determined by g).

Definition 3. A *vector space* over complex numbers is set V, equipped with *scalar multiplication* and *vector addition*, which satisfies the following axioms: For every $c, c_1, c_2 \in \mathbb{C}$ and for every $\boldsymbol{x}, \boldsymbol{x}_1, \boldsymbol{x}_2 \in V$,

1. $\boldsymbol{x}_1 + \boldsymbol{x}_2 = \boldsymbol{x}_2 + \boldsymbol{x}_1$.
2. $(\boldsymbol{x}_1 + \boldsymbol{x}_2) + \boldsymbol{x} = \boldsymbol{x}_1 + (\boldsymbol{x}_2 + \boldsymbol{x})$.
3. $\mathbf{0} + \boldsymbol{x} = \boldsymbol{x} + \mathbf{0} = \boldsymbol{x}$.
4. For any $\boldsymbol{x}$, there exists an *additive inverse* $-\boldsymbol{x}$ such that $\boldsymbol{x} + (-\boldsymbol{x}) = \mathbf{0}$.
5. $c_1(c_2\boldsymbol{x}) = (c_1c_2)\boldsymbol{x}$.
6. $(c_1 + c_2)\boldsymbol{x} = c_1\boldsymbol{x} + c_2\boldsymbol{x}$.
7. $c(\boldsymbol{x}_1 + \boldsymbol{x}_2) = c\boldsymbol{x}_1 + c\boldsymbol{x}_2$.

8. $1\boldsymbol{x} = \boldsymbol{x}$.

The elements of V are called *vectors*.

Definition 4. Let V and W be two vector spaces. A *linear map* from V to W is a function $f : V \to W$ such that for all $\boldsymbol{x}, \boldsymbol{x}_1, \boldsymbol{x}_2 \in V$ and $c \in \mathbb{C}$,

1. $f(\boldsymbol{x}_1 + \boldsymbol{x}_2) = f(\boldsymbol{x}_1) + f(\boldsymbol{x}_2)$,
2. $f(c \cdot \boldsymbol{x}) = c \cdot f(\boldsymbol{x})$.

We call any linear map from a vector space to itself a *linear operator*.

Definition 5. Let V be a vector space. A *linear combination* of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in V$ is a finite sum $c_1\boldsymbol{x}_1 + \ldots + c_n\boldsymbol{x}_n$, where $c_1, \ldots, c_n \in \mathbb{C}$. A set $S \subset V$ is called *linearly independent* if

$$c_1\boldsymbol{x}_1 + \ldots + c_n\boldsymbol{x}_n = \boldsymbol{0}$$

implies $c_1 = \cdots = c_n = 0$ whenever $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in S$. This means that the only way that a linear combination of vectors can be zero is if all the c_i's are zero. A set that is not linearly independent is *linearly dependent*.

Definition 6. Let V be a vector space and let $W = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}\} \subset V$ be a set. We say that W is a *basis* for V if the following two conditions are satisfied:

1. Every $\boldsymbol{x} \in V$ can be written as a linear combination of vectors from W, i.e. in this case we say that W *spans* V.
2. W is linearly independent.

The number of elements in W gives the *dimension* of W. Throughout this course we will only consider finite dimensional complex vector spaces.

A natural way to introduce geometry into a complex vector space V is to define the inner product.

Definition 7. Let V be a complex vector space. An *inner product* on V is a mapping $V \times V \to \mathbb{C}$, denoted as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$, that satisfies the following conditions for any $c_1, c_2 \in \mathbb{C}$ and any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ in V.

1. $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle^*$.
2. $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0$ if and only if $\boldsymbol{x} = \boldsymbol{0}$.

3. $\langle \boldsymbol{x}, c_1\boldsymbol{y} + c_2\boldsymbol{z}\rangle = c_1\langle \boldsymbol{x}, \boldsymbol{y}\rangle + c_2\langle \boldsymbol{x}, \boldsymbol{z}\rangle$.

A vector space equipped with an inner product is also called an *inner product space*.

For vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1 \ldots, y_n)$ in $\mathbb{C}^n$, the formula

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle = x_1^* y_1 + \ldots + x_n^* y_n$$

defines the inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Definition 8. Two vectors are called *orthogonal* if their inner product is 0.

From trigonometry, we understand that, if V and V' are unit vectors, the inner product $\langle V, V'\rangle$ is the directional length of the projection of V onto the direction of V'. The inner product can be geometrically interpreted in that way as we shall be dealing with unit vectors.

Definition 9. Let C be an $n \times n$ matrix. The *trace* of C, denoted by Trace(C), is the sum of its diagonal elements. That is,

$$\text{Trace}(C) = \sum_{i=0}^{n-1} C[i, i].$$

Definition 10. We define the *norm* or the *length* of a vector V as

$$||V|| = \sqrt{\langle V, V\rangle}.$$

If $||V|| = 1$, then V is called a *unit vector*.

Definition 11. A set V of vectors is called *orthonormal* if every vector in V is pairwise orthogonal and is of unit length.

Definition 12. A complex inner product space V is *complete* if for any Cauchy sequence of vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$, there exists a vector $\boldsymbol{y} \in V$ such that

$$\lim_{n\to\infty} |\boldsymbol{x}_n - \boldsymbol{y}| = 0.$$

The intuition behind this is that a vector space with an inner product is complete if any sequence accumulating somewhere converges to a point.

Definition 13. A *Hilbert space* is a complex inner product space that is complete.

It is worth noting that every finite dimensional complex inner product space is automatically complete. Hence any finite dimensional complex inner product space is a Hilbert space. Since we only work with finite dimensional vector spaces in quantum computing, we do not have to worry about the completeness criteria.

Definition 14. For a matrix A in $\mathbb{C}^{n\times n}$, if there is a number $c \in \mathbb{C}$ and a vector $V \in \mathbb{C}^n$ such that

$$AV = cV$$

then c is called an *eigenvalue* of A and V is called an *eigenvector* of A associated to c.

Every eigenvector determines a complex vector subspace of the vector space. This space is known as the *eigenspace* associated with the given eigenvector.

Definition 15. Let A be an $n \times n$ matrix. Define the *adjoin* of A to be $A^\dagger = (A^T)^*$, where A^* denotes the complex conjugate of A. The matrix A is called *unitary* if $AA^\dagger = A^\dagger A = I_n$, where I_n is the $n \times n$ identity matrix.

1.4.1 Combining Systems with Tensor Product

We will often deal with multiple systems and we will want to combine them as a whole single compound system. Tensor product is used to combine two or more systems into one. We will be using tensor products to represent combined quantum systems.

Definition 16. The *tensor product* of $r \times s$ and $t \times u$ matrices

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1s} \\ a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \dots & a_{rs} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1u} \\ b_{21} & b_{22} & \dots & b_{2u} \\ \vdots & \vdots & \ddots & \vdots \\ b_{t1} & b_{t2} & \dots & b_{tu} \end{bmatrix},$$

is an $rt \times su$ matrix defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1s}B \\ a_{21}B & a_{22}B & \dots & a_{2s}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1}B & a_{r2}B & \dots & a_{rs}B \end{bmatrix}$$

To give an example how a combined system works, consider the marble example we gave in the beginning of this chapter. Imagine now a blue marble placed on a graph whose vertices are labeled as a, b, c and whose adjacency matrix is given as

$$M = \begin{bmatrix} 0 & \frac{1}{6} & \frac{5}{6} \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{6} \\ \frac{2}{3} & \frac{1}{3} & 0 \end{bmatrix}$$

Imagine also a red marble placed on another separate graph whose vertices are labeled as $0, 1$ and whose adjacency matrix is defined as

$$N = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

Suppose that we want to combine these two systems. Then,

$$M \otimes N = \begin{bmatrix} 0 \cdot N & \frac{1}{6} \cdot N & \frac{5}{6} \cdot N \\ \frac{1}{3} \cdot N & \frac{1}{2} \cdot N & \frac{1}{6} \cdot N \\ \frac{2}{3} \cdot N & \frac{1}{3} \cdot N & 0 \cdot N \end{bmatrix}$$

would be the dynamics of the combined system represented by a 6×6 matrix.

A state in the combined two-marble system would be the tensor product of the state of the blue marble system and the state of the red marble system. Since the blue marble can be in one of three possible state and the red marble can be in one of two possible states, the tensor product of the combined system has six possible states. Let

$$B = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

be a state in a blue marbled system and let

$$R = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

be a state in a red marbled system. The state for the combined system would look like as follows:

$$X = \begin{matrix} a0 \\ a1 \\ b0 \\ b1 \\ c0 \\ c1 \end{matrix} \begin{bmatrix} \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \end{bmatrix},$$

which corresponds to the fact that there is a

$\frac{1}{6}$ chance of the blue marble being on vertex a and the red marble being on vertex 0.

$\frac{1}{6}$ chance of the blue marble being on vertex a and the red marble being on vertex 1.

$\frac{1}{6}$ chance of the blue marble being on vertex b and the red marble being on vertex 0.

$\frac{1}{6}$ chance of the blue marble being on vertex b and the red marble being on vertex 1.

$\frac{1}{6}$ chance of the blue marble being on vertex c and the red marble being on vertex 0.

$\frac{1}{6}$ chance of the blue marble being on vertex c and the red marble being on vertex 1.

2 Basic Quantum Theory

We shall cover in this section how quantum states are formally represented and what kind of postulates quantum systems have.

2.1 Qubits

From classical computing we know that a *bit* is a boolean value that can either be in the basis state 0 or 1, but not both. A *quantum bit (qubit)* is the basic information content for an intended quantum system. A qubit can be in the basis state 0 or 1 or simultaneously both. We denote these *basis states* by $|0\rangle$ and $|1\rangle$.[1] This is called the *ket* notation, invented by the famous physicist Paul Dirac.[2] We shall denote the quantum states by lowercase Greek letters like ψ, ϕ, φ.

A general state of a qubit $|\psi\rangle$ is a linear combination (superposition) of states in the canonical basis. That is,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$.

The coefficients α and β are called *amplitudes*. So $|\psi\rangle$ can be denoted by a two dimensional complex vector

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

such that $|\alpha|^2 + |\beta|^2 = 1$. Note that when $\alpha = 1$ and $\beta = 0$, we have the basis state

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

When $\alpha = 0$ and $\beta = 1$, we get the basis state

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

So essentially a qubit can be viewed as a unit length vector in a two dimensional Hilbert space. We will see shortly, though, whenever we measure a qubit, it immediately collapses into a classical bit.

[1] We call this set of basis states, the *canonical basis*.

[2] It is also called the *Dirac notation* for this reason.

How are qubits implemented? There are many ways to achieve this. A qubit can be represented by one of the following phenomena.

1. Two possible states of an electron around the nucleus, being in the *ground state* or the *excited state*.
2. A photon having two different polarizations, i.e. *horizontal* and *vertical* polarizations.
3. A subatomic particle having two energy levels.

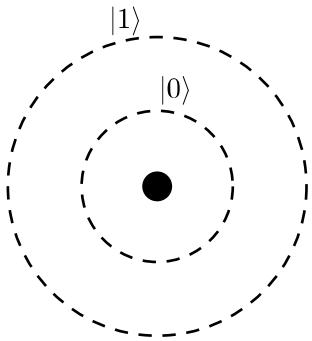


Figure 6: The state of an electron. We may let $|0\rangle$ denote the ground state and let $|1\rangle$ denote the excited state.

The examples of course are not limited to those we listed above.

Geometric representation. A nice way to represent qubits is by the polar form. Recall that given a complex number c,

$$|c|^2 = c \times c^* = (a + bi) \times (a - bi) = a^2 + b^2.$$

Given a complex number $c = a + bi$ of modulus 1, we can visualize it as an arrow of unit length from the origin of the circle of unit radius.

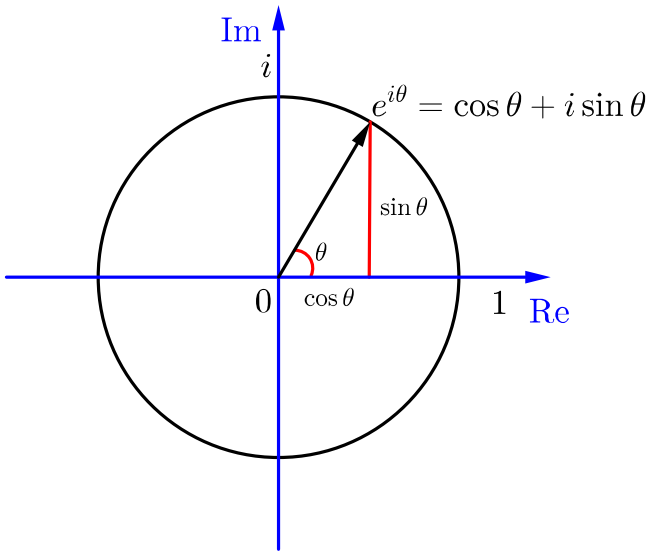


Figure 7: Trigonometric representation of a complex number.

As seen in Figure 7, every complex number can be identified by an angle θ that the vector makes with the positive x axis and a magnitude ρ (we may omit ρ if we only consider vectors in the unit circle). A complex number c in its general form, using polar coordinates, therefore can be written as

$$c = \rho(\cos\theta + i\sin\theta).$$

The following formula is known as *Euler's formula* and it is expressed as

$$e^{i\theta} = \cos\theta + i\sin\theta.$$

Note that when $\theta = \pi$, it follows that $e^{\pi i} = -1$. So using Euler's formula we can write c simply as

$$c = \rho e^{i\theta}.$$

There is an analogous representation of a qubit as an arrow from the origin to a three dimensional ball. Now a generic qubit is of the form

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle,$$

where $|c_0|^2 + |c_1|^2 = 1$. Although it may look like there are four real number parameters in this equation, it turns out that there are only two parameters. Since

$$c_0 = r_0 e^{i\phi_0}$$

and

$$c_1 = r_1 e^{i\phi_1},$$

the generic form of a qubit can be rewritten as

$$|\psi\rangle = r_0 e^{i\phi_0}|0\rangle + r_1 e^{i\phi_1}|1\rangle.$$

Defining r_0 as $\cos\theta$ and r_1 as $\sin\theta$, and using the fact that the squared modulus of the amplitudes add up to 1, any qubit $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$ can be then represented as

$$|\psi\rangle = \cos(\theta)|0\rangle + e^{i\phi}\sin(\theta)|1\rangle.$$

Since only the relative phase between the coefficients of the two basis vectors has any physical meaning, we can take the coefficient of $|0\rangle$ to be real and non-negative representation. When we only care about the relative phases of the states but not their actual phases, this gives rise to *Bloch sphere*.

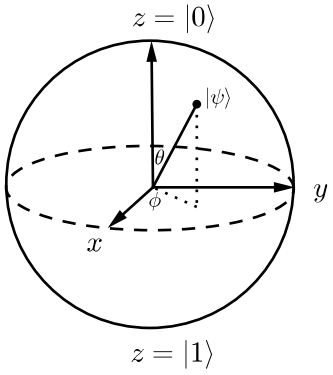


Figure 8: Bloch sphere representation of a single qubit quantum system $|\psi\rangle$.

A qubit can be represented by a unit length vector in a *Bloch sphere* as shown in Figure 8. We just need two angles that describe such a vector in the Bloch sphere. A qubit $|\psi\rangle$ then can be written as

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

The angle θ shows the *latitude* and ϕ shows the *longitude (phase)*. The precise meaning of the above equation is the following: ϕ is the angle that

$|\psi\rangle$ makes from x axis along the equator and θ is the angle that $|\psi\rangle$ makes with the z axis. When a qubit is measured in the standard basis, it collapses to a bit, or equivalently, to the north or south pole of the Bloch sphere.

Rotating a vector around the z axis is changing its phase. Notice that the probability of which classical state it will collapse to is not affected. Such transformation is called a *phase change*. In the equation given above, it corresponds to altering the phase parameter $e^{i\phi}$.

Single qubit systems may not be very practical. It is not possible to solve a reasonable problem with single qubit systems. We will need quantum computing systems involving multiple qubits. This is obtained by tensor products which was introduced in the last section.

Notation. Considering the definition of tensor products, let $|\psi\rangle \in \mathbb{C}^n$, $|\psi'\rangle \in \mathbb{C}^m$. Then $\mathbb{C}^n \otimes \mathbb{C}^m = \mathbb{C}^{n\times m}$ and $|\psi\rangle \otimes |\psi'\rangle = |\psi\rangle|\psi'\rangle = |\psi, \psi'\rangle = |\psi\psi'\rangle$. It is worth noting that the tensor product of vectors does not commute. For example, $|1\rangle|0\rangle \neq |0\rangle|1\rangle$.

As an example to a tensor product, suppose that we are given a state $|\varphi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, and a state $|\psi\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$. The tensor product of these two single qubit systems is

$$|\varphi\rangle \otimes |\psi\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} \end{bmatrix}.$$

So the tensor product is a four dimensional (two qubit) system with basis states $00, 01, 10, 11$. Then, the two qubit system above is

$$|\varphi\rangle \otimes |\psi\rangle = \frac{1}{2\sqrt{2}}|00\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|01\rangle + \frac{1}{2\sqrt{2}}|10\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|11\rangle.$$

Then, a 2-qubit system would have $2^2 = 4$ states in total, i.e. $00, 01, 10, 11$. These canonical basis states can be represented by the vectors as, respec-

tively,

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

A state of a 2-qubit system in a four dimensional Hilbert space H^4 has the form

$$|\psi\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle = \begin{bmatrix}c_0\\c_1\\c_2\\c_3\end{bmatrix},$$

where $|c_0|^2 + |c_1|^2 + |c_2|^2 + |c_3|^2 = 1$.

Generally, we may have n-dimensional quantum systems, generalizing our earlier implementation example:

1. A particle that can be in one of n positions.
2. A system that might have one of n energy levels.
3. A photon might have one of n polarization directions.

A state of an n-dimensional quantum system $|\psi\rangle$ is represented by a unit length vector in an n-dimensional Hilbert space H^n

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle + \ldots + c_{n-1}|n-1\rangle = \begin{bmatrix}c_0\\c_1\\\vdots\\c_{n-1}\end{bmatrix}$$

as a linear combination of n many mutually orthogonal basis states $|0\rangle, |1\rangle, \ldots, |n-1\rangle$ such that

$$\sum_{i=0}^{n-1} |c_i|^2 = 1.$$

Entanglement. Multiple qubit systems exhibit a very fundamental quantum mechanical phenomenon called *entanglement* that was not present in the double-slit experiment. Consider a two qubit system

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$$

such that $\sum |\alpha_i|^2 = 1$. The probability to see $|\psi\rangle$ in state $|00\rangle$ is $|\alpha_0|^2$, to see in state $|01\rangle$ is $|\alpha_1|^2$, and so on. What about if we only measure the first qubit? This is called a *partial measurement*. What is the result of measuring just the first qubit? The probablity of seeing $|0\rangle$ in the first qubit is simply $|\alpha_0|^2 + |\alpha_1|^2$. A more interesting question is the following: If the outcome of the first qubit is 0, what is the new state after the measurement? Well, if the outcome of the first qubit is 0, it is certainly not $|10\rangle$ and $|11\rangle$ because these states are inconsistent with the outcome. What is left is the state $|00\rangle + |01\rangle$. Of course this state is not normalized now so we need to normalize it. The new state then becomes

$$|\psi'\rangle = \frac{\alpha_0|00\rangle + \alpha_1|01\rangle}{\sqrt{|\alpha_0|^2 + |\alpha_1|^2}}.$$

For example, let us look at the state

$$|\psi\rangle = \left(\frac{1}{2} + \frac{i}{2}\right)|00\rangle + \frac{1}{2}|01\rangle + \frac{i}{2}|11\rangle.$$

What is the probability of seeing $|0\rangle$ in the first qubit? It is simply $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$. The new state is

$$|\psi'\rangle = \frac{\left(\frac{1}{2} + \frac{i}{2}\right)|00\rangle + \frac{1}{2}|01\rangle}{\sqrt{\frac{3}{4}}}.$$

Recall that to combine two systems we use the tensor product. Given two qubits, we apply tensor product to get a combined 2-qubit system. What about the inverse process? Given a 2-qubit system, can we always factorize it into two systems of single qubit so that their tensor product is the composite 2-qubit system? It turns out that this is not always the case.

A state that is not the tensor product of the smaller states is called an *entangled state*. Consider 2-qubit systems. A state $z \in H^4$ of a two qubit system is *decomposable* if z can be written as a product of states in H_2, $z = x \otimes y$. A state that is not decomposable is *entangled*, and in this case the qubits are entangled to each other.

Example. The state $\dfrac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}$ is decomposable, since

$$\tfrac{1}{2}(|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle) = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

On the other hand the state $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ is entangled.[3] Suppose the contrary that

$$\begin{aligned}\tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &= (a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle) \\ &= a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle\end{aligned}$$

for some complex numbers a_0, a_1, b_0, b_1. But then

$$\begin{aligned}a_0b_0 &= \tfrac{1}{\sqrt{2}} \\ a_0b_1 &= 0 \\ a_1b_0 &= 0 \\ a_1b_1 &= \tfrac{1}{\sqrt{2}}.\end{aligned}$$

So a_0b_0 and a_1b_1 are non-zero. Then, a_0, b_0, a_1, b_1 are all non-zero. But then neither a_0b_1 nor a_1b_0 can be zero. A contradiction.

What happens when we measure the Bell state? We will see the state $|00\rangle$ with probability $\frac{1}{2}$ and see $|11\rangle$ with the same probability. What is important here is that if we measure the first qubit, the second qubit turns out exactly the same with the first qubit, and vice versa. Imagine we prepare a quantum state in the Bell state and send one qubit to Mars and leave the other in Earth. Since two qubits are entangled, whenever we apply an operation on one qubit, the same operation will be automatically applied to the other.

2.2 State evolution

We said that quantum systems are probabilistic. A state in a two dimensional quantum system, say, will look like

$$|\psi\rangle = \begin{bmatrix}\alpha_0 \\ \alpha_1\end{bmatrix}$$

First of all let us take a look at the state $|\psi\rangle$. We should not interpret $|\psi\rangle$ like a state of a probabilistic system. It is incorrect to say by looking at $|\psi\rangle$ that the probability of a particle being in position k is $|\alpha_k|^2$. In a quantum system, to be in state $|\psi\rangle$ means that the particle is in *all* positions simultaneously. It is in a *superposition* of basis states. Now if we perform a *measurement* the superposition collapses to a single classical state. $|\psi\rangle$ says,

[3] This state is called a *Bell state* and it will be widely used in quantum information exchange protocols. We will also call it an *EPR pair*.

after measuring the system, the particle will be found in position k with probability $|\alpha_k|^2$.

Geometrically, state evolution can be seen as a rotation of the vector space. This rotation is represented by a matrix. However, we require the transformation to be unitary.

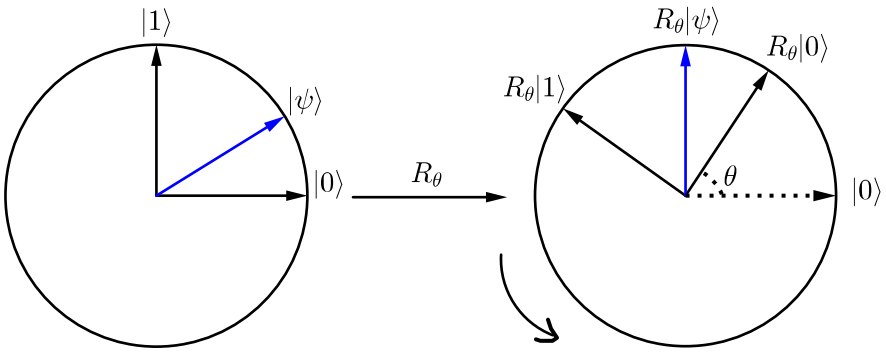


Figure 9: State evolution is a rotation of the vector space.

In the figure above, the rotation matrix maps the state $|0\rangle$ to state $\cos\theta|0\rangle + \sin\theta|1\rangle$, and it maps the state $|1\rangle$ to state $-\sin\theta|0\rangle + \cos\theta|1\rangle$. This rotation matrix can be then defined as

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

The inverse of this rotation is just the transpose of R_θ. That is,

$$R_\theta^{-1} = R^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

Note that $R_\theta R_\theta^{-1} = R_\theta^{-1} R_\theta = I$.

In quantum computing, we require that all transformations are unitary. The first reason why transformations must be unitary is because quantum mechanical processes are *reversible*. Reversible in the sense that we are able to uniquely obtain the input from the output. Unitary operators are reversible since their inverse is simply their adjoin. Given a unitary operator U and a vector V, we have that

$$V \to UV \to U^\dagger UV \to IV = V.$$

Second reason for using unitary matrices is that they preserve inner products and the norm of the vector.

Proposition 17. Unitary operators preserve inner products. That is, if U is a unitary matrix, then for any V and V' in $\mathbb{C}^n$ we have $\langle UV, UV' \rangle = \langle V, V' \rangle$.

Proof. The proof follows directly from the definition of unitary operators.

$$\langle UV, UV' \rangle = (UV)^\dagger (UV') = (V^\dagger U^\dagger)(UV') = V^\dagger V' = \langle V, V' \rangle.$$

The first and fourth equalities follow from the property of inner product, the second equality follows from the adjoin operator over matrices, the third equality follows from the fact that $U^\dagger U$ gives the identity matrix since U is unitary. □

Corollary 18. Unitary operators preserve norms, i.e., $||UV|| = ||V||$.

Proof. It is easy to observe that

$$||UV|| = \sqrt{\langle UV, UV \rangle} = \sqrt{\langle V, V \rangle} = ||V||.$$

The first and third equalities follow from the definition of the norm, the second equality follows from the previous proposition. □

In this course we consider state evolution of quantum systems. However in a typical quantum mechanics course, one learns that the continuous time evolution of a closed quantum system (ignoring special relativity) follows the *Schrödinger equation*

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = \hat{H}(t)|\psi(t)\rangle,$$

where $\hbar$ is a physical constant known as *Planck's constant* and $\hat{H}(t)$ is a Hermitian operator known as the *Hamiltonian* of the system. The Hamiltonian is an operator which represents the total energy function for the system. It may be a function of time but it may also be taken as constant.

2.3 Observables and measurement

In order to extract quantum information from a quantum system we have to observe the system, i.e. perform a *measurement*. We shall only consider projection measurements in our study for now.

Suppose that we are given a qubit

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle,$$

where $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Any measurement on $|\psi\rangle$ will cause the qubit to decide which one of the basis states it will remain in.

A measurement is therefore a *projection* onto one of the basis states $|i\rangle$ with probability $|\alpha_i|^2$, where $i \in \{0, 1\}$.

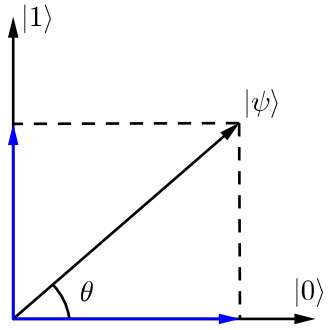


Figure 10: Measuring a qubit is a projection onto one of the basis states.

When we measure the state $|\psi\rangle$ we get $|0\rangle$ with probability $\cos^2\theta$ or get $|1\rangle$ with probability $\sin^2\theta$.

We may in fact measure a state in an any set of basis states which are mutually orthogonal. Take for example the basis states

$$|u\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$|u^\perp\rangle = -\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

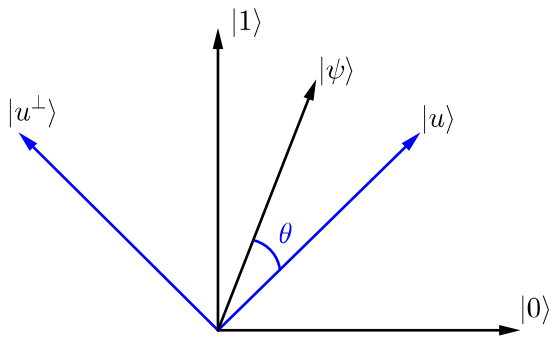


Figure 11: Change of basis.

If we measure $|\psi\rangle$ in the $\{|u\rangle, |u^\perp\rangle\}$ basis, we get $|u\rangle$ with probability $\cos^2\theta$, or get $|u^\perp\rangle$ with probability $\sin^2\theta$. In fact, the probability of seeing $|\psi\rangle$ in the $|u\rangle$ basis state is determined by the square of the inner product of $|\psi\rangle$ and $|u\rangle$.

Measuring in an arbitrary basis provides us to write any state $|\psi\rangle$ in that basis. Consider the *sign basis* which we shall define as

$$|+\rangle = \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle,$$
$$|-\rangle = \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle.$$

This is a basis where $|+\rangle$ is the vector with a $\frac{\pi}{4}$ degrees angle with the state $|0\rangle$, and $|-\rangle$ is the vector which is the reflection of $|+\rangle$ around the state $|0\rangle$. So it should look like as follows

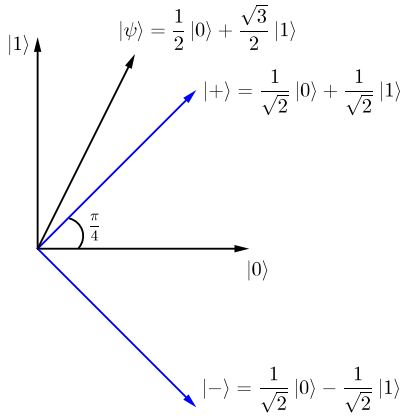


Figure 12: Representation of the sign basis.

Notice that these vectors are normalized and they are orthogonal to each other as their inner product is zero. Now after measuring, the probability of seeing $|\psi\rangle$ in state $|+\rangle$ is basically the square of the inner product of $|\psi\rangle$ and $|+\rangle$, which is

$$\left(\frac{1}{2}\cdot\frac{1}{\sqrt{2}} + \frac{\sqrt{3}}{2}\cdot\frac{1}{\sqrt{2}}\right)^2 = \frac{2+\sqrt{3}}{4}.$$

How do we write $|\psi\rangle$ in the sign basis?

$$\begin{aligned}
|\psi\rangle &= \alpha|+\rangle + \beta|-\rangle \\
|0\rangle &= \tfrac{1}{\sqrt{2}}|+\rangle + \tfrac{1}{\sqrt{2}}|-\rangle \\
|1\rangle &= \tfrac{1}{\sqrt{2}}|+\rangle - \tfrac{1}{\sqrt{2}}|-\rangle \\
|\psi\rangle &= \tfrac{1}{2}\left(\tfrac{1}{\sqrt{2}}|+\rangle + \tfrac{1}{\sqrt{2}}|-\rangle\right) + \tfrac{\sqrt{3}}{2}\left(\tfrac{1}{\sqrt{2}}|+\rangle - \tfrac{1}{\sqrt{2}}|-\rangle\right) \\
&= \left(\tfrac{1+\sqrt{3}}{2\sqrt{2}}\right)|+\rangle + \left(\tfrac{1-\sqrt{3}}{2\sqrt{2}}\right)|-\rangle
\end{aligned}$$

Observables. We now turn to question, what do we measure? We measure observables. An *observable* is a quantity like energy, momentum, position.

Definition 19. An $n \times n$ matrix A is called *Hermitian* if $A^\dagger = A$. If A is a Hermitian matrix then the operator that it represents is called *self-adjoint*.

Example. The matrix

$$\begin{bmatrix} 5 & 4+5i & 6-16i \\ 4-5i & 13 & 7 \\ 6+16i & 7 & -2.1 \end{bmatrix}$$

is Hermitian.

In quantum mechanics, given an n-dimensional quantum system, an observable is an $n \times n$ Hermitian operator.

By definition, the diagonal entries of a Hermitian operator must be real. This is due to the fact that if the diagonal entries were complex, then it would have to be $c = c^*$. But if $c = c^*$, then c must be a real number.

Let us remind that the following property about the inner product that if $V, V' \in \mathbb{C}^n$, then

$$\langle V, V' \rangle = \langle V', V \rangle^*.$$

Proposition 20. If A is an $n \times n$ Hermitian matrix, then for all V and V' in $\mathbb{C}^n$, we have that

$$\langle AV, V' \rangle = \langle V, AV' \rangle.$$

Proof. The proof is easy. We have

$$\langle AV, V' \rangle = (AV)^\dagger V' = V^\dagger A^\dagger V' = V^\dagger A V' = \langle V, AV' \rangle$$

The first and fourth equalities follow from the definition of an inner product. The second equality is from the property of $\dagger$ in matrix operations. The third equality follows from the fact that A is a Hermitian matrix, i.e., $A^\dagger = A$. □

It is imporant to note that the eigenvalues of a Hermitian matrix are all real numbers.

Proposition 21. If A is a Hermitian matrix, then all eigenvalues of A are real.

Proof. Suppose that A is a Hermitian matrix with eigenvalue λ and a nonzero eigenvector V. We show that $\lambda \langle V, V \rangle = \lambda^* \langle V, V \rangle$.

$$\lambda^* \langle V, V \rangle = \langle \lambda V, V \rangle = \langle AV, V \rangle = \langle V, AV \rangle = \langle V, \lambda V \rangle = \lambda \langle V, V \rangle.$$

The first and fifth equalities follow from the property of the inner product. The second and fourth properties follow from the definition of eigenvalue. The third equality follows from the last proposition. Now since λ and V are non-zero and $\lambda = \lambda^*$, then λ must be a real number. $\square$

Moreover, we have the following proposition.

Proposition 22. For a given Hermitian matrix A, distinct eigenvectors of A which have distinct eigenvalues are orthogonal.

Proof. Let V_1 and V_2 be distinct eigenvectors of a Hermitian matrix A.

$$AV_1 = c_1V_1 \quad \text{and} \quad AV_2 = c_2V_2.$$

Then we have the following equalities:

$$c_1\langle V_1, V_2\rangle = c_1^*\langle V_1, V_2\rangle = \langle c_1V_1, V_2\rangle = \langle AV_1, V_2\rangle = \langle V_1, AV_2\rangle$$

$$= \langle V_1, c_2V_2\rangle = c_2\langle V_1, V_2\rangle.$$

The first and last equalities follow from the fact that eigenvalues of Hermitian matrices are real. The second equality is from the properties of inner product, the third and fifth equalities are by definition of eigenvector, the fourth equality follows from the fact that A is Hermitian. As the leftmost expression is equal to the rightmost expression, we may subtract one from the other to get 0. That is,

$$c_1\langle V_1, V_2\rangle - c_2\langle V_1, V_2\rangle = (c_1 - c_2)\langle V_1, V_2\rangle = 0.$$

Because c_1 and c_2 are distinct, $c_1 - c_2 \neq 0$. Hence, it follows that $\langle V_1, V_2\rangle = 0$. Therefore, V_1 and V_2 are orthogonal since their inner product is 0. $\square$

The orthogonality has a physical meaning. By *Heisenberg's uncertainity principle*, we can only measure either the position or the momentum of an electron of, say, a hydrogen atom. We cannot measure both at the same time.

Definition 23. A *diagonal matrix* is a square matrix whose only non-zero entries are on the diagonal.

Theorem 24 (Spectral Theorem)**.** Every self-adjoint operator A on a finite-dimensional complex vector space V can be represented by a diagonal matrix whose diagonal entries are the eigenvalues of A, and whose eigenvectors form an orthonormal basis for V which we shall call it an *eigenbasis*.

Spectral theorem says that every Hermitian operator A has orthonormal eigenvectors $|\phi_1\rangle \ldots, |\phi_n\rangle$ with real eigenvalues $\lambda_1 \ldots, \lambda_n$ such that $A|\phi_i\rangle = \lambda_i|\phi_i\rangle$. So Hermitian operators and the spectral theorem are really just mathematical formalisms for specifying orthonormal basis for measurements. If we have a general state

$$|\psi\rangle = \alpha_1|\phi_1\rangle + \cdots + \alpha_n|\phi_n\rangle,$$

measuring a Hermitian A on $|\psi\rangle$ is a projection onto one of the eigenvectors of A and the outcome will be the corresponding eigenvalue λ_j (a real number) with probability $|\alpha_j|^2$. After the measurement the new state will be $|\phi_j\rangle$. How we read the eigenvalue λ_j can be imagined as having a needle on a measurement device deflecting the number λ_j.

Example. Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ be a single qubit quantum state and let

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

be an observable. Suppose that we want to make a measurement with respect to the observable X. We need to determine the eigenvalues and eigenvectors of X. Fortunately, they are quite simple. The eigenvectors are $|+\rangle$ and $|-\rangle$. The corresponding eigenvalues are 1 and -1, respectively. We can verify this easily.

$$X|+\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

For the other eigenvector,

$$X|-\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = -1 \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

So how do we make the measurement? We want to write $|\psi\rangle$ in the basis of the eigenvectors. In this case, in the sign basis. Remembering what $|0\rangle$ and $|1\rangle$ are in the sign basis, we get

$$|\psi\rangle = \left(\frac{\alpha+\beta}{\sqrt{2}}\right)|+\rangle + \left(\frac{\alpha-\beta}{\sqrt{2}}\right)|-\rangle.$$

Now if we make a measurement on $|\psi\rangle$, we get the eigenvalue 1 with probability $\left|\frac{\alpha+\beta}{\sqrt{2}}\right|^2$ and the new state in this case will be $|+\rangle$; Or we get the

eigenvalue -1 with probability $\left|\frac{\alpha-\beta}{\sqrt{2}}\right|^2$ and the new state in this case will be $|-\rangle$.

What if we have repeated eigenvalues? It may be the case that we have the same eigenvalues for different eigenvectors. Let us have a state $|\psi\rangle$ in a 3-dimensional space with a given observable, say a Hermitian operator A, with eigenvalues $\lambda_1, \lambda_2, \lambda_3$ with the corresponding orthonormal eigenvectors $|\phi_1\rangle, |\phi_2\rangle, |\phi_3\rangle$.

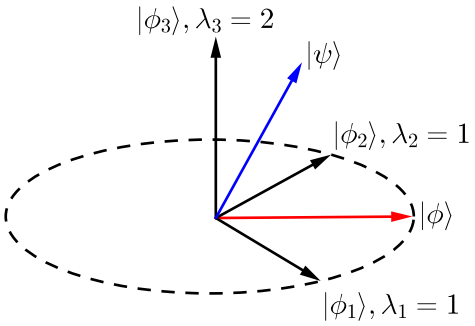


Figure 13: Repeated eigenvalues.

This won't be a problem, because any vector $|\phi\rangle$ that is a linear combination of $|\phi_1\rangle$ and $|\phi_2\rangle$ will also be an eigenvector with the same eigenvalue 1. That is,

$|\phi\rangle = \alpha|\phi_1\rangle + \beta|\phi_2\rangle$. If we apply A on $|\phi\rangle$, we get
$A|\phi\rangle = \alpha A|\phi_1\rangle + \beta A|\phi_2\rangle$
$= \alpha \cdot 1|\phi_1\rangle + \beta \cdot 1|\phi_2\rangle$
$= \alpha|\phi_1\rangle + \beta|\phi_2\rangle$
$= |\phi\rangle$.

We now want to generalize the definition of observables for measuring in an arbitrary basis $|\phi_1\rangle, \ldots |\phi_n\rangle$ with arbitrary real outcomes $\lambda_1, \ldots, \lambda_n$. We can define an observable with corresponding eigenvectors and eigenvalues. Let us demonstrate this with an example.

Suppose that the eigenvectors are

$$|\phi_1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle, \qquad |\phi_2\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{i}{\sqrt{2}}|1\rangle.$$

with corresponding eigenvalues $\lambda_1 = 1$ and $\lambda_2 = -1$ respectively. We now define a Hermitian operator A having these eigenvectors and eigenvalues. When we are given a state $|\psi\rangle$ we want to project it onto a state $|\phi\rangle$. The way we do this is by the *projection matrix* P defined as

$$P = |\phi\rangle\langle\phi|,$$

where $\langle\phi| = |\phi\rangle^\dagger$ is called the *dual* of $|\phi\rangle$. So P projects a given state onto $|\phi\rangle$. It is easy to verify this as we can check that

$$P|\psi\rangle = |\phi\rangle\langle\phi||\psi\rangle = |\phi\rangle(\langle\phi||\psi\rangle)$$

The paranthesis is just the inner product of $|\phi\rangle$ and $\psi\rangle$ so it gives us a complex number as a coefficient of $|\phi\rangle$. We may then write the last expression as

$$P|\psi\rangle = (\langle\phi||\psi\rangle)|\phi\rangle$$

Now we claim that we can define the observable A as follows

$$\begin{aligned} A &= \lambda_1|\phi_1\rangle\langle\phi_1| + \lambda_2|\phi_2\rangle\langle\phi_2| \\ &= 1\cdot|\phi_1\rangle\langle\phi_1| + (-1)\cdot|\phi_2\rangle\langle\phi_2| \end{aligned}$$

It can be verified that A is a Hermitian matrix with eigenvectors $|\phi_1\rangle$ and $|\phi_2\rangle$ with eigenvalues λ_1 and λ_2, respectively associated with the eigenvectors. Let us verify that $A|\phi_1\rangle = \lambda_1|\phi_1\rangle$.

$$\begin{aligned} A|\phi_1\rangle &= (|\phi_1\rangle\langle\phi_1| - |\phi_2\rangle\langle\phi_2|)|\phi_1\rangle \\ &= |\phi_1\rangle\langle\phi_1||\phi_1\rangle - |\phi_2\rangle\langle\phi_2||\phi_1\rangle \\ &= |\phi_1\rangle(\langle\phi_1||\phi_1\rangle) - |\phi_2\rangle(\langle\phi_2||\phi_1\rangle) \\ &= |\phi_1\rangle\cdot 1 - |\phi_2\rangle\cdot 0 \\ &= |\phi_1\rangle \end{aligned}$$

So $A|\phi_1\rangle = \lambda_1|\phi_1\rangle$. We leave the reader to verify that $A|\phi_2\rangle = \lambda_2|\phi_2\rangle$. We can also verify this by explicitly writing the matrix A.

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{i}{\sqrt{2}} \end{bmatrix} + (-1) \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & -\frac{i}{2} \\ \frac{i}{2} & \frac{1}{2} \end{bmatrix} + (-1) \begin{bmatrix} \frac{1}{2} & \frac{i}{2} \\ -\frac{i}{2} & \frac{1}{2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & -\frac{i}{2} \\ \frac{i}{2} & \frac{1}{2} \end{bmatrix} + \begin{bmatrix} -\frac{1}{2} & -\frac{i}{2} \\ \frac{i}{2} & -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

From this we can show that $A|\phi_2\rangle = \lambda_2|\phi_2\rangle$. This is simply,

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix} = (-1) \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} \end{bmatrix}.$$

So to generalize all this, given an orthonormal set of vectors $\{|\phi_1\rangle, \ldots, |\phi_n\rangle\}$ and set of eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$, the corresponding observable is

$$A = \sum \lambda_i |\phi_i\rangle\langle\phi_i|$$

In summary, the two notions of measuring quantum systems are equivalent.

Now that we know the properties of Hermitian operators, we can give the following postulates of quantum computing.

Postulate. To each physical observable of a quantum system there corresponds a Hermitian operator.

Example. Let $|\psi\rangle = \left[\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}i}{2}\right]^T$ be the start state in the two-dimensional state space. Let

$$\Omega = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & -i \\ i & 1 \end{bmatrix}$$

This matrix acts as an operator on $\mathbb{C}^2$. Therefore, we can apply it to $|\psi\rangle$. The result is the vector $\Omega|\psi\rangle = [-1, 0]^T$. Observe that $|\psi\rangle$ and $\Omega|\psi\rangle$ are *not* scalar multiples of one another, and thus they do not represent the same state: Ω has modified the state of the system.

Postulate. The eigenvalues of a Hermitian operator Ω associated with a physical observable are the only possible values the observable can take as

a result of measuring it on any given state. Furthermore, eigenvectors of Ω form a basis for the state space.

Observables can be thought of as legitimate questions as we can pose to quantum systems. Each question admits a set of answers: The eigenvalues of the observable.

In classical physics we assume that measuring a system would leave the system in the same state. This was shown to be wrong in quantum physics. The observable can only assume one of its eigenvalues as the result of an observation. Nothing tells us how frequently we are going to see a specific eigenvalue λ. Moreover our framework does not tell us yet what happens to the state vector if λ is actually observed. We introduce the following postulate.

Postulate. Let Ω be an observable and $|\psi\rangle$ be a state. If the result of measuring Ω on state $|\psi\rangle$ is the eigenvalue λ, the state after measurement will always be an eigenvector corresponding to λ.

Consider the same observable Ω given in the previous example. Let the eigenvalues of Ω be λ_1 and λ_2, and let the corresponding eigenvectors be $|e_1\rangle$ and $|e_2\rangle$.

Now, let us suppose that after an observation of Ω on a state $|\psi\rangle = \frac{1}{\sqrt{2}}[1,1]^T$, the actual value observed is λ_1. The system has "collapsed" from $|\psi\rangle$ to $|e_1\rangle$.

If we are working with non-normalised states, recall that if $A|\phi\rangle = c|\phi\rangle$ for some c then $c|\phi\rangle$ represents physically the same state as $|\phi\rangle$. So if the system is in an eigenvector of the basis, then the system will not change.

2.4 Density Matrices and Mixed States

Given a quantum system, a *pure state* is described as a superposition

$$|\psi\rangle = \sum_i \alpha_i |i\rangle,$$

where $\alpha_i \in \mathbb{C}$ such that $\sum_i |\alpha_i|^2 = 1$. In many cases, a quantum state may lose its coherence and stability due to environmental interference or

uncontrollable factors. In such cases, the information of the relative phase in the quantum state may get lost. For this reason we instead consider a statistical ensemble of pure states, called *mixed state*. Hence, we may think of a mixed state as a probability distribution of pure states. Mixed states, as well as pure states, can be represented by density matrices. Let $|\psi_i\rangle$ be a set of pure states and let p_i be the probability of the system being in state $|\psi_i\rangle$ such that $0 \leq p_i \leq 1$ and $\sum_i p_i = 1$. Then, we write the corresponding density matrix as

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|.$$

Example 25. Suppose as a probability distribution that a quantum state is in the $|0\rangle$ basis state with probability $1/2$, and in the $|1\rangle$ state with probability $1/2$. Then,

$$|0\rangle\langle 0| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} [1, 0] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$

$$|1\rangle\langle 1| = \begin{bmatrix} 0 \\ 1 \end{bmatrix} [0, 1] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

The corresponding mixed state is then defined as

$$\rho = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

Assume that we are given a set of pure states $\{p_i, |\psi_i\rangle\}$, where p_i denotes the probability of the quantum system being in state $|\psi_i\rangle$. Suppose that ρ is the mixed state of the system. A measurement on ρ with respect to the basis states $\{|\beta_l\rangle\}$ will produce the state $|\beta_l\rangle$ as an outcome with probability $\langle\beta_l|\rho|\beta_l\rangle$. In fact, when measuring the mixed state generated by the pure states $\{p_i, |\psi_i\rangle\}$, with respect to the standard basis, the probability of obtaining $|\beta_l\rangle$ is equal to the diagonal entry $\rho[l, l]$ of the density matrix of the mixed state.

Some of the properties of density matrices can be given as follows:

(i) $\rho^2 = \rho$ if and only if ρ is the density matrix of a pure state.

(ii) $\rho^\dagger = \rho$.

(iii) $\text{Trace}(\rho) = 1$.

(iv) Every eigenvalue of ρ is non-negative.

The *expectation value* of an observable A is defined as

$$\begin{aligned}
\langle A \rangle &= \sum_j p_j \langle \psi_j | A | \psi_j \rangle \\
&= \sum_j p_j \text{Trace} \left(|\psi_j\rangle\langle\psi_j| A \right) \\
&= \sum_j \text{Trace} \left(p_j |\psi_j\rangle\langle\psi_j| A \right) \\
&= \text{Trace} \left(\sum_j p_j |\psi_j\rangle\langle\psi_j| A \right) \\
&= \text{Trace}(\rho A)
\end{aligned}$$

For our purpose, we will consider pure states in our study. That is, we assume that the quantum systems we have are closed systems in the sense that they are isolated from the environment sufficiently enough to the extent that the qubits evolve with no physical interruption. This of course may not be a realistic expectation, as electrons may get entangled uncontrollably with the other electrons surrounding them. In any occurance of "error", we may lose the information of the relative phase of the qubit. The difference between pure states and mixed states is actually the loss of relative phase information.

2.5 EPR Paradox

Quantum theory was a breakthrough discovery in physics. However Einstein neither liked nor really believed in the quantum theory that Niels Bohr and himself devised. In reaction to the uncertainity in quantum mechanics, he even claimed that "God does not play dice", meaning that the universe is determined by well defined physical laws. Of course Einstein's concept of God was not transcendental, but he rather regarded God as the natural force behind the laws of the universe. To undermine the quantum theory, Einstein, Podolsky, and Rosen presented a thought experiment that was claimed to conflict with the principles of quantum mechanics.

Suppose that we are given a Bell state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

that we want to measure in the sign basis. We want to know the probability of seeing the state $|+\rangle$ when measuring the first qubit and if we see a plus, we want to know the probability of seeing $|+\rangle$ in the second qubit. To know

this we need to write the Bell state in the sign basis. We claim that the Bell state in the sign basis is actually again a Bell state. That is, we claim that

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle = \frac{1}{\sqrt{2}}|++\rangle + \frac{1}{\sqrt{2}}|--\rangle.$$

To see this, write the right handside as

$$\tfrac{1}{\sqrt{2}}\left((\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle)(\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle) + (\tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle)(\tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle)\right)$$

which can be written as

$$\tfrac{1}{\sqrt{2}}\left((\tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle) + (\tfrac{1}{2}|00\rangle - \tfrac{1}{2}|01\rangle - \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle)\right)$$

If we cancel the terms, we get

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle.$$

So the probability we see a plus in the first qubit is just $\frac{1}{2}$. The new state in this case will become $|++\rangle$. So the probability of seeing a plus in the second qubit after measuring the first qubit is 1. The EPR (Einstein-Podolsky-Rosen) paradox is that, unlike Heisenberg's uncertainity principle, we may know the bit value and the sign value of the first qubit at the same time. What we do is we prepare an entangled state and move the second qubit to a distant galaxy or planet. We measure the first qubit in the standard canonical basis. At the same time we also measure the second qubit in the sign basis. Since these qubits are physically separated from each other and since the light does not have a chance to travel from the second qubit to the first qubit, measuring the second qubit will not disturb the state of the first qubit. Measuring the second qubit we get the sign value. This contradicts Heisenberg's uncertainity principle. The way quantum mechanics explains this paradox is that as soon as we measure the first qubit, the state will not be entangled anymore. If we, say, measure the first qubit as $|0\rangle$ in the canonical basis, the new state will be $|00\rangle$. But then this new state is not entangled anymore since it can be written as the tensor product of two single-qubit states $|0\rangle|0\rangle$.

In fact given any orthonormal basis states $|u\rangle$ and $|u^\perp\rangle$, the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ in the $\{|u\rangle, |u^\perp\rangle\}$ basis is still a Bell state $\frac{1}{\sqrt{2}}|uu\rangle + \frac{1}{\sqrt{2}}|u^\perp u^\perp\rangle$. This property is called the *rotational invariance of Bell state*.

Exercise. Show that the rotational invariance property holds for the basis states

$$|u\rangle = a|0\rangle + b|1\rangle, \quad |u^{\perp}\rangle = -a|0\rangle + b|1\rangle,$$

where assume for simplicity $a, b \in \mathbb{R}$ and $a^2 + b^2 = 1$.

Continuing our discussion, let us suppose that we want to measure the second qubit in a different orthonormal basis $\{|v\rangle, |v^{\perp}\rangle\}$.

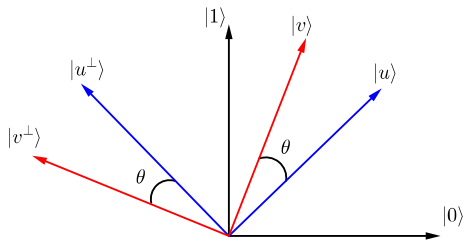


Figure 14: Rotational invariance of Bell state.

If the outcome of the first qubit is $|u\rangle$ then the new state of the system will be $|uu\rangle$. Now if we measure the second qubit in basis $\{|v\rangle, |v^{\perp}\rangle\}$, the probability of seeing $|v\rangle$ is simply $\cos^2\theta$. Similarly, if it if the outcome of measuring the first qubit was $|u^{\perp}\rangle$, the new state would be $|u^{\perp}u^{\perp}\rangle$ since we are working with the Bell state, and the probability of seeing $|v^{\perp}\rangle$ would be again $\cos^2\theta$. So the probability of seeing orthogonal complements is preserved through when working with the entangled Bell state.

2.6 Classical gates and quantum gates

Now we look at how to implement classical and quantum gates via matrices. An gate with n input bits and m output bits is represeted by an $2^m \times 2^n$ matrix. We represent n input bits as a $2^n \times 1$ matrix and m output bits as a $2^m \times 1$ matrix. So a $2^m \times 2^n$ matrix takes a $2^n \times 1$ matrix and outputs a $2^m \times 1$ matrix. Consider the NOT gate. Classical NOT gate takes a single bit and outputs a single bit. Fortunately, it is same for qubits. NOT of $|0\rangle$ equals $|1\rangle$. NOT of $|1\rangle$ equals $|0\rangle$.

$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The ith column of the matrix denotes the output of the function given the ith basis state as an input. So the first column of NOT shows what the output will be for the first input case (for $|0\rangle$ that is), and the second column shows what the output will be for the second input case (for $|1\rangle$). Hence, note that

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

and also

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

What about other classical gates? Here are the descriptions of other gates used. Each will be a 2×4 matrix.

$$AND = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, NAND = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$OR = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}, NOR = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$XOR = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

2.6.1 Reversible computation.

Quantum computation is heavily related to reversible computation since the time evolution of states in quantum mechanics is unitary and so it is inherently reversible. This reversibility is a consequence of restricting attention to closed quantum systems (ignoring special relativity). A computation is *reversible* if the input can be uniquely recovered from the output. The NOT gate for example is reversible since it is its own inverse. On the other hand, AND gate is not reversible.

In non-reversible gates, for example in AND gate, we lose information. Given the output 0, we may not know what the original input was. So we lose that information. Interestingly, quantum computation was first studied to try to understand whether or not the reversibility requirement of quantum mechanics impose any limit on what we can compute. It was not clear in the

1970's if we could compute anything in classical computing quantumly. Every step of a computation must dissapate some amount of energy. But this energy is not due to producing information but it is due to losing information. This was came to known as *Landauer's principle*. Fortunately enough, it was shown that any classical computation can be done reversibly and so, in theory, no energy or heat would be produced. So any non-reversible gate can be transformed into a reversible gate, computing the same function (we will introduce Toffoli gates in a moment for this purpose).

We want our quantum gates to be reversible. Moreover, we do not want any *junk* information in the output of our quantum circuit for reasons which will be clear in a moment. Given a classical function f, we want to define it quantumly. At any stage of the computation we should be able to recover the computation and compute the inverse function. But one problem is the case when we try to find a reversible counterpart for functions $f : \{0,1\}^n \to \{0,1\}^m$ such that $m < n$. For example the AND gate. But that is just one example. How can we recover the computation if we lose all the information?

To do the computation reversibly, we need to introduce additional auxiliary input and output wires, preserving equal number of input and output wires. We obtain a reversible version of a given irreversible circuit, by simply replacing all the non-reversible components with their reversible counterparts (as any gate can be made reversible). If we start with the output, and run the circuit backwards by replacing each gate by its inverse, we obtain the input again. So the reversible version of a non-reversible gate might require a constant number of additional wires. If the gate has a different number of input and output bits, then it cannot be reversible unless we modify it by adding the additional wires. However, adding auxiliary wires will generate *junk* bits along with the output (denoted by j in Figure 15). Can't we just ignore the junk bits? Unfortunately, we cannot do this since the output may be entangled with the junk bits. To demonstrate this, assume we start with the state $|x_1\rangle + |x_2\rangle$ (let us ignore the scalar amplitudes for simplicity). Then the reversible version of the function, $|x\rangle|0\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle|j(x)\rangle$, applied on a superposition $|x_1\rangle + |x_2\rangle$ will give the output

$$|x_1\rangle|f(x_1)\rangle|j(x_1)\rangle + |x_2\rangle|f(x_2)\rangle|j(x_2)\rangle$$

If we apply, say the Hadamard transform (see *Quantum Gates* subsection)

on the first qubit, that is, if we apply the mapping

$$|x_1\rangle \mapsto \frac{|x_1\rangle + |x_2\rangle}{\sqrt{2}}, \quad |x_2\rangle \mapsto \frac{|x_1\rangle - |x_2\rangle}{\sqrt{2}},$$

we end up with (ignoring the scalars again)

$$(|x_1\rangle + |x_2\rangle)|f(x_1)\rangle|j(x_1)\rangle + (|x_1\rangle - |x_2\rangle)|f(x_2)\rangle|j(x_2)\rangle.$$

If $f(x_1) = f(x_2)$ and $j(x_1) = j(x_2)$, then $|x_2\rangle$ will completely cancel out, i.e. they interfere. But what if it were the case that $f(x_1) = f(x_2)$ but $j(x_1) \neq j(x_2)$? Then, there would be no interference in the above expression. But there would definitely be an interference if we could get rid of the junk bits. So in this case, having extra junk bit makes a difference between not having it.

The additional junk information generated by making each gate reversible can be erased at the end of the computation by first copying the output, and then running the inverse of the function to obtain the start state again. The general schema of a reversible circuit for quantum computation will look something along the lines of the circuit given in Figure 15.

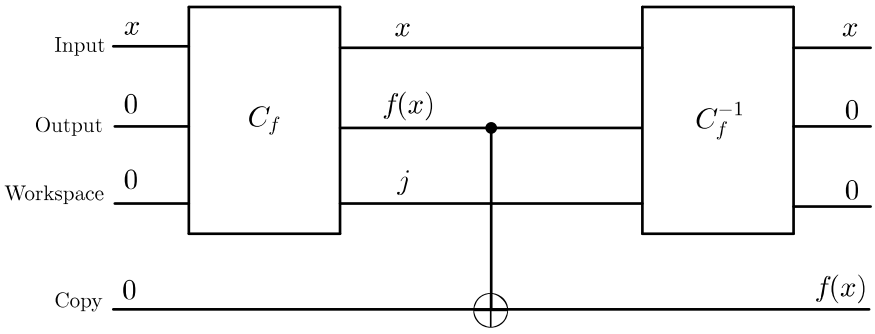


Figure 15: General schema of a reversible circuit for quantum computation. C_f denotes the classical reversible counterpart of a given irreversible function f.

The above schema is for reversibly (in fact quantumly) computing a given function f. The block labelled C_f represents a circuit composed of reversible gates for simulating the function f. We copy the output $f(x)$ to another register and then run the circuit C_f^{-1} (inverse of C_f) to erase the contents

of the output and the junk information. Notice that if we did not copy the output bit to another register, we would not have the output in our hand. Having the above entire circuit as our reversible version of the function f that can be taken as a quantum circuit. The output of this circuit is x, $f(x)$ and a bunch of workspace qubits. We got rid of the junk information as we promised. Erasing the junk information was essential because junk output prevents interference.

Let us introduce the *controlled-NOT* (CNOT) gate which is defined as follows.

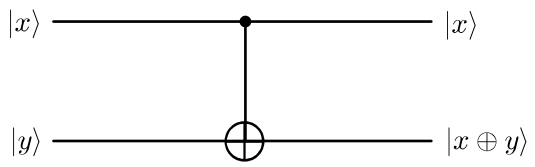


Figure 16: CNOT gate.

The CNOT gate has two inputs and two outputs. The top input is the control bit. It controls the output. It inverts the second input only when the control bit is active. So if $|x\rangle = |0\rangle$, then $|y\rangle$ remains the same. It inverts $|y\rangle$ only when $|x\rangle = |1\rangle$. Then CNOT gate takes $|x, y\rangle$ to $|x, x \oplus y\rangle$, where $\oplus$ is the binary *exclusive-or* (XOR) operation. The truth table of $CNOT$ is given as follows.

x	y	$CNOT(x,y)$	
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

The matrix representation of CNOT is

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

It can be easily verified that CNOT is unitary. Hence, the CNOT operation is reversible. Observe that

$$CNOT(CNOT(|A, B\rangle)) = |A, B\rangle.$$

Another reversible gate is called the *Toffoli gate* which is defined as follows.

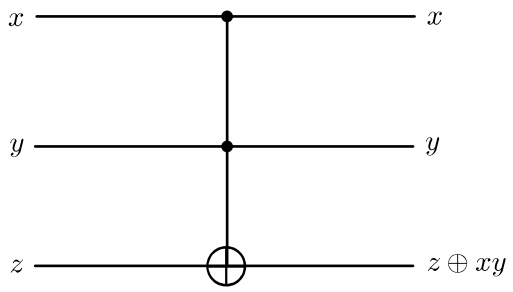


Figure 17: Toffoli gate.

The first two bits are used for controlling the target bit. So, in the boolean algebraic form, we have

$$T(x_1,\ x_2,\ x_3) = (x_1,\ x_2,\ x_1x_2 + x_3).$$

Toffoli gate is a reversible gate. Moreover, it is *universal* for classical computation. Another universal gate for classical computation is the NAND gate which is defined as $NOT(x \wedge y)$. Any classical logic gate can be simulated solely by using Toffoli gates. For example we can simulate the AND gate using only Toffoli gates. Notice that $T(x_1, x_2, 0) = (x_1, x_2, x_1x_2)$. So using 0 in the third input, Toffoli gates computes the logical AND of x_1 and x_2. Now about simulating other gates, since $x_1 \vee x_2 = \neg(\neg x_1 \wedge \neg x_2)$, we can replace all OR gates by AND and NOT gates. We know that NOT gate is reversible and we just showed that the AND gate can also be reversible using Toffoli gate. So in principle, classical computation can be solely made reversible.

Quantum gates. We also have gates which are used exclusively for quantum computing. One of the most important is called the *Hadamard transform*. Single qubit Hadamard transform is defined as

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Hadamard transformation basically creates a superposition of basis states. For example, when we apply H on the basis state $|0\rangle$, we get

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

Verify that the Hadamard transform is unitary. Note that it is its own inverse. So if we apply it twice we get the original input. Therefore, the Hadamard gate is reversible.

What Hadamard transform does is that it rotates the vector space around the $\pi/8$ axis 180 degrees. So, the basis state $|0\rangle$ gets mapped to $|+\rangle$, and $|1\rangle$ gets mapped to $|-\rangle$.

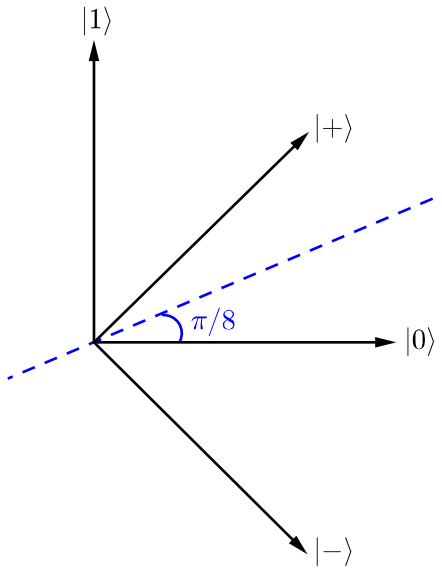


Figure 18: Hadamard transform is a rotation of the vector space around the $\pi/8$ axis.

Another important one bit gates are called *Pauli gates* which correspond to rotation around the x, y and z axes of the Bloch sphere. Pauli gates are defined as follows:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Here, note that X is nothing different than the NOT gate and it corresponds to the rotatation of the given vector around the $\pi/4$ axis. Also note that the X, Y, Z Pauli gates rotate the Bloch sphere 180° about the x, y, z axes respectively. Z is also known as the *phase flip* (or phase change) gate. Sometimes we may want to turn vector not by 180° around an axis but just about θ degrees along a particular direction. When applied for the Z-gate, such transformation would become a *phase shift* gate defined as

$$R(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix},$$

for some real θ. Note that $R(\theta)$ leaves $|0\rangle$ the same but changes the phase parameter of $|1\rangle$.

What if we want to apply the Hadamard transform to two qubit system. In this case, we apply the matrix

$$H^{\otimes 2} = H \otimes H.$$

So for an n-qubit system we take the tensor product of the gate n times and apply the resulting matrix to the system. What if we want to apply different gates to the qubits? If we want to apply, say, the NOT gate and the Hadamard gate to the first and second qubits, respectively, we take the tensor product of the NOT gate and the Hadamard gate $X \otimes H$ and apply this matrix to our system.

We spoke a lot about the Bell state so we should also give the Bell state circuit and how to create it. To create a Bell state all we need is a Hadamard transform and a CNOT gate providing the input $|00\rangle$.

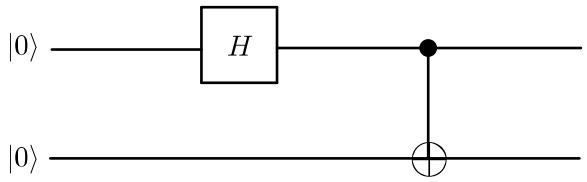


Figure 19: Bell state circuit.

We start with the state $|00\rangle$. We apply the Hadamard transform on the first qubit and get

$$(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)|0\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle.$$

Now if we apply the CNOT gate taking the second qubit as our target qubit and taking the first qubit as our control qubit, we get

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

which is how we get the Bell state. In fact, we get all the orthonormal set of Bell states when given different inputs $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. As an exercise we leave the reader to work with these inputs.

Let us review the postulates of quantum systems. We summarize the postulates as follows.

State Space Postulate. The state of an n-dimensional quantum system is described by a unit length vector in an n-dimensional Hilbert space.

Evolution Postulate. The state evolution of a closed quantum system is described by a unitary operator. That is, for any evolution of the closed system there exists a unitary operator U such that if the initial state of the system is $|\psi_1\rangle$, then the state of the system after the evolution will be

$$|\psi_2\rangle = U|\psi_1\rangle.$$

Geometrically, each unitary transformation can be seen as a rotation of the vector or the vector space.

Composition of Systems Postulate. When two physical systems, say H_1 and H_2, are treated as one combined system, the state space of the

combined physical system is the tensor product space $H_1 \otimes H_2$ of the state spaces H_1 and H_2 of the component subsystems. If the first system is in the state $|\psi_1\rangle$ and the second system in the state $|\psi_2\rangle$, then the state of the combined system is

$$|\psi_1\rangle \otimes |\psi_2\rangle.$$

Measurement Postulate. For a given orthonormal basis $\mathcal{B} = \{|\phi_i\rangle\}$ of a state space H_A for a system A, it is possible to perform a *measurement* on system A with respect to basis $\mathcal{B}$ that, given a state

$$|\psi\rangle = \sum_i \alpha_i |\phi_i\rangle,$$

outputs a label i with probability $|\alpha_i|^2$ and leaves the system in state $|\phi_i\rangle$. Furthermore, given a state $|\psi\rangle = \sum_i \alpha_i |\phi_i\rangle|\gamma_i\rangle$ from a bipartite state space $H_1 \otimes H_2$ (every $|\phi_i\rangle$ is mutually orthogonal; the $|\gamma_i\rangle$ have unit norm but are not necessarily orthogonal), then performing a measurment on system A will yield outcome i with probability $|\alpha_i|^2$ and leave the bipartite system in state $|\phi_i\rangle|\gamma_i\rangle$.

3 Quantum Teleportation

Quantum teleportation is the process by which the state of an arbitrary qubit is transferred from one location to another. As we shall see however, when the state of a qubit is teleported to another location, the state of the original qubit will have to be necessarily destroyed. The teleportation protocol will be heavily based on the use of the entanglement of EPR pairs.

Note that if two qubits are in an entangled state of $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, then observing one of them will give 0 or 1, both with probability of $\frac{1}{2}$, but it is not possible to observe different values of the qubits. This quantum correlation can remain even if the qubits are seperated from each other light years away. The correlation itself plays an important role in quantum communication protocols. The state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is called en *EPR pair* (also called a Bell state). Note that if we apply the Hadamard matrix on an EPR pair, the result is again EPR pair.

3.1 No-Cloning Theorem

Definition 26. Let $\{|a_1\rangle, \ldots, |a_n\rangle\}$ be a set of basis states. Let us denote the state space by H_n and specify that the state $|a_1\rangle$ is a "blank sheet state". A unitary mapping in $H_n \otimes H_n$ is called a *quantum copy machine*, if for any state $|\psi\rangle \in H_n$,

$$U(|\psi\rangle|a_1\rangle) = |\psi\rangle|\psi\rangle.$$

Theorem 27 (No Cloning Theorem)**.** For $n > 1$, there is no quantum copy machine.

Proof. Assume that a quantum copy machine U exists for $n > 1$. Since $n > 1$, there are two orthogonal states $|a_1\rangle$ and $|a_2\rangle$. We should have $U(|a_1\rangle|a_1\rangle) = |a_1\rangle|a_1\rangle$ and $U(|a_2\rangle|a_1\rangle) = |a_2\rangle|a_2\rangle$. Also, if we let the first input qubit of U to be a superposition, we have that

$$U(\tfrac{1}{\sqrt{2}}(|a_1\rangle + |a_2\rangle)|a_1\rangle) = (\tfrac{1}{\sqrt{2}}(|a_1\rangle + |a_2\rangle))(\tfrac{1}{\sqrt{2}}(|a_1 + |a_2\rangle))$$

$$= \frac{1}{2}(|a_1\rangle|a_1\rangle + |a_1\rangle|a_2\rangle + |a_2\rangle|a_1\rangle + |a_2\rangle|a_2\rangle).$$

But since U is linear,

$$U(\tfrac{1}{\sqrt{2}}(|a_1\rangle + |a_2\rangle)|a_1\rangle) = \tfrac{1}{\sqrt{2}}U(|a_1\rangle|a_1\rangle) + \tfrac{1}{\sqrt{2}}U(|a_2\rangle|a_1\rangle)$$

$= \frac{1}{\sqrt{2}}|a_1\rangle|a_1\rangle + \frac{1}{\sqrt{2}}|a_2\rangle|a_2\rangle.$

The two representations above for $U(\frac{1}{\sqrt{2}}(|a_1\rangle + |a_2\rangle)|a_1\rangle)$ do not coincide by the very definition of a tensor product. A contradiction. □

So copying an arbitrary quantum state is not possible unless we destroy it, i.e. only cut-and-paste is allowed.

3.2 No-Deleting Theorem

We now give the time reversed dual of the cloning theorem. Given two copies of some arbitrary quantum state, we ask if it is possible to delete one of the copies. Due to linearity of quantum mechanics this turns out to be impossible. The deletion that is considered here is not the same as classical irreversible erasure but is more like reversible *uncopying* of an unknown quantum state.

Mathematically, a quantum deleting process takes two copies of an arbitrary quantum state as input and outputs a blank state together with the original input. More precisely,

$$U|\psi\rangle|\psi\rangle|A\rangle = |\psi\rangle|\Sigma\rangle|A'\rangle,$$

where U is the deleting operation which is linear, $|\psi\rangle$ is the unknown state, $|\Sigma\rangle$ is the blank state, $|A\rangle$ is the initial ancilla state of the deleting machine, and $|A'\rangle$ is the final state of the machine. The following theorem is due to Pati and Braunstein (2000).

Theorem 28 (No-Deleting Theorem)**.** Let $|\psi\rangle$ be an unknown quantum state. Then, there is no linear isometric transformation such that

$$|\psi\rangle|\psi\rangle|A\rangle \to |\psi\rangle|\Sigma\rangle|A'\rangle$$

with the final state of the ancilla being independent of $|\psi\rangle$.

Proof. The quantum deletion process involves two initially identical qubits (e.g. photons of arbitrary polarization) in some state $|\psi\rangle$ and an ancilla in some state $|A\rangle$. The action is defined by

$$|\psi\rangle|\psi\rangle|A\rangle \to |\psi\rangle|\Sigma\rangle|A_\psi\rangle \tag{1}$$

where $|A_\psi\rangle$ is the final state of the ancilla, which may in general depend on the polarization of the original photon. An obvious solution to this equation

is to swap the second and third states. However, this would be same as the standard erasure process where the extra copies have played no role. We therefore explicitly exclude swapping as describing quantum deleting. We consider the transformation given above on a pair of horizontally and vertically polarized photons:

$$|H\rangle|H\rangle|A\rangle \rightarrow |H\rangle|\Sigma\rangle|A_H\rangle$$

$$|V\rangle|V\rangle|A\rangle \rightarrow |V\rangle|\Sigma\rangle|A_V\rangle. \tag{2}$$

In fact, transformation (1) defines a whole class of possible deleting machines which could behave differently if the two inputs are unequal or even entangled, e.g.,

$$\frac{1}{\sqrt{2}}(|H\rangle|V\rangle + |V\rangle|H\rangle)|A\rangle \rightarrow |\Phi\rangle \tag{3}$$

where $|\Phi\rangle$ might be any state of the combined input-ancilla system. Now for an arbitrary input qubit $|\psi\rangle = \alpha|H\rangle + \beta|V\rangle$ and for arbitrary complex amplitudes α and β such that $|\alpha|^2 + |\beta|^2 = 1$, the linearity and the transformations (2) and (3) of the deleting machine yields

$$\begin{array}{ll} & |\psi\rangle|\psi\rangle|A\rangle \\ = & (\alpha|H\rangle + \beta|V\rangle)(\alpha|H\rangle + \beta|V\rangle)|A\rangle \\ = & (\alpha^2|H\rangle|H\rangle + \alpha\beta|H\rangle|V\rangle + \alpha\beta|V\rangle|H\rangle + \beta^2|V\rangle|V\rangle)|A\rangle \\ = & \left[\alpha^2|H\rangle|H\rangle + \beta^2|V\rangle|V\rangle + \alpha\beta(|H\rangle|V\rangle + |V\rangle|H\rangle)\right]|A\rangle \\ \rightarrow & \alpha^2|H\rangle|\Sigma\rangle|A_H\rangle + \beta^2|V\rangle|\Sigma\rangle|A_V\rangle + \sqrt{2}\alpha\beta|\Phi\rangle \end{array} \tag{4}$$

This is a quadratic polynomial in α and β. However, if (1) was to hold, then the yield in (4) must reduce to

$$(\alpha|H\rangle + \beta|V\rangle)|\Sigma\rangle|A_\psi\rangle \tag{5}$$

for all α and β. But in general, the yield in (4) and (5) are not identical, hence we say that the machine fails to delete a copy. If we require the output states to be the same, then since $|\Phi\rangle$ is independent of α and β and so $|A_\psi\rangle$ is linear with them, the only solution is when

$$|\Phi\rangle = \frac{1}{\sqrt{2}}(|H\rangle|\Sigma\rangle|A_V\rangle + |V\rangle|\Sigma\rangle|A_H\rangle)$$

and

$$|A_\psi\rangle = \alpha|A_H\rangle + \beta|A_V\rangle.$$

Since the final state (5) must be normalized for all possible α and β, it follows that the ancilla states $|A_H\rangle$ and $|A_V\rangle$ are orthogonal. This means that the quantum information is simply in the final state $|\Phi\rangle$ of the ancilla. Thus, the transformation (1) is not uncopying at all, but merely swapping onto a two-dimensional subspace of the ancilla. One can always obtain the unknown state from the final state of the ancilla using local operation on the ancilla Hilbert space. It appears that there is no option but to move the information around without deleting it. Hence, linearity of quantum theory does not allow an unknown quantum state to be deleted perfectly. □

3.3 Teleportation protocol

Consider two communication parties Alice (denoted by A) and Bob (denoted by B). Suppose that Alice has a single qubit in state

$$a|0\rangle + b|1\rangle$$

which is actually unknown to Alice. Suppose that Alice wishes to send this state to Bob. We say that there is a *quantum channel* from Alice to Bob if it is possible to send Bob the whole two-state quantum system. Similarly, if Alice can send classical bits to bob, we say that there is a *classical channel* from Alice to Bob.

We assume that there is no quantum channel from Alice to Bob, but assume a classical channel exists. Alice cannot measure her state since it would disturb her qubit. She cannot make copies of her qubit either for many observations since copying an arbitrary quantum state is not possible due to No Cloning Theorem. It seems that it is possible for Alice to send her quantum bit to bob by only using a classical channel.

On the other hand, if Alice and Bob initially share an EPR pair, there is a way to execute the required task. This protocol is called *quantum teleportation*. Suppose that Alice and Bob have two qubits in EPR state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Assume that the first qubit belongs to Alice and the second qubit belongs to Bob. In addition, Alice has her qubit to be teleported in state

$$a|0\rangle + b|1\rangle$$

The compound state of all of the qubits will be denoted as

$(a|0\rangle + b|1\rangle)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

$= \frac{a}{\sqrt{2}}|0\rangle|00\rangle + \frac{a}{\sqrt{2}}|0\rangle|11\rangle + \frac{b}{\sqrt{2}}|1\rangle|00\rangle + \frac{b}{\sqrt{2}}|1\rangle|11\rangle$

$= \frac{a}{\sqrt{2}}|000\rangle + \frac{a}{\sqrt{2}}|011\rangle + \frac{b}{\sqrt{2}}|100\rangle + \frac{b}{\sqrt{2}}|111\rangle.$

Now recall that only the third qubit belongs to Bob. Alice can access the first two qubits.

Teleportation protocol.

1. Alice performs the CNOT matrix on her qubits, using the first qubit as the control qubit. The 3-qubit compound state then becomes

$$\frac{a}{\sqrt{2}}|000\rangle + \frac{a}{\sqrt{2}}|011\rangle + \frac{b}{\sqrt{2}}|110\rangle + \frac{b}{\sqrt{2}}|101\rangle$$

2. Next, Alice applies the Hadamard matrix on the first qubit. The result is

$\frac{a}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|00\rangle + \frac{a}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|11\rangle$
$+\frac{b}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|10\rangle + \frac{b}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|01\rangle$

which can be written as

$\frac{a}{2}|000\rangle + \frac{a}{2}|100\rangle + \frac{a}{2}|011\rangle + \frac{a}{2}|111\rangle$
$+\frac{b}{2}|010\rangle - \frac{b}{2}|110\rangle + \frac{b}{2}|001\rangle - \frac{b}{2}|101\rangle$

Taking apart Bob's qubit, last state can be rewritten as

$\frac{1}{2}|00\rangle a|0\rangle + \frac{1}{2}|10\rangle a|0\rangle + \frac{1}{2}|01\rangle a|1\rangle + \frac{1}{2}|11\rangle a|1\rangle$
$+\frac{1}{2}|01\rangle b|0\rangle - \frac{1}{2}|11\rangle b|0\rangle + \frac{1}{2}|00\rangle b|1\rangle - \frac{1}{2}|10\rangle b|1\rangle.$

and, moreover as,

$\frac{1}{2}|00\rangle(a|0\rangle + b|1\rangle) + \frac{1}{2}|01\rangle(a|1\rangle + b|0\rangle)$
$+\frac{1}{2}|10\rangle(a|0\rangle - b|1\rangle) + \frac{1}{2}|11\rangle(a|1\rangle - b|0\rangle).$

3. Now Alice observes her two qubits. As the outcome, she sees $00, 01, 10, 11$, each with probability of $\frac{1}{4}$.

Alice's obervation	Post-observation state
00	$\|00\rangle(a\|0\rangle + b\|1\rangle)$
01	$\|01\rangle(a\|1\rangle + b\|0\rangle)$
10	$\|10\rangle(a\|0\rangle - b\|1\rangle)$
11	$\|11\rangle(a\|1\rangle - b\|0\rangle)$

Recall that the first two qubits belong to Alice, and the third qubit belongs to Bob.

4. Alice sends Bob her observation result (two classical bits).
5. Bob performs the following

- If Alice's bits are 00, Bob makes nothing. His qubit is already in state $a|0\rangle + b|1\rangle$.
- If Alice sent 01, Bob performs NOT gate on his qubit $a|1\rangle + b|0\rangle$ and get the desired state $a|0\rangle + b|1\rangle$.
- If Alice sent 10, Bob performs the *phase-flip* matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

  on his qubit to get $a|0\rangle + b|1\rangle$.
- If Alice sent 11, Bob first performs NOT gate and then phase-change gate.

Remark: It is worth noting that in quantum teleportation, no physical qubit is transmitted, just the state of the qubit. The quantum state is transmitted, not copied. So the original state held by Alice is destroyed in the protocol.

3.4 Superdense Coding

Superdense coding is the complementary action to quantum teleportation. Initially, Alice and Bob share an EPR pair and there is *quantum channel* from Alice to Bob. Now Alice wants to send *classical bits* to Bob. Superdense

coding is a protocol where Alice sends one qubit to Bob, but the amount of transmitted information is two classical bits. This is done as follows: Alice has two classical bits, b_1 and b_2, and Alice shares an EPR pair, with Bob,

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

We assume that the first qubit belongs to Alice, and the other one belongs to Bob.

Superdense coding protocol

1. If $b_1 = 1$, then Alice performs the phase-change gate on her qubit. If $b_2 = 1$, then she also performs the NOT gate on her qubit.

b_1 b_2	State after Alice's operation
0 0	$\frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$
0 1	$\frac{1}{\sqrt{2}}(\|10\rangle + \|01\rangle)$
1 0	$\frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$
1 1	$\frac{1}{\sqrt{2}}(\|10\rangle - \|01\rangle)$

2. Alice sends her qubit to Bob.

3. Now that Bob has access to both qubits, he runs them both through a two-qubit gate B defined as

$$B = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}$$

Note that B is a unitary matrix. It is easy then to verify the following table

b_1 b_2	State after Alice's operation	State after Bob's operation
0 0	$\frac{1}{\sqrt{2}}\|00\rangle + \|11\rangle$	$\|00\rangle$
0 1	$\frac{1}{\sqrt{2}}\|10\rangle + \|01\rangle$	$\|01\rangle$
1 0	$\frac{1}{\sqrt{2}}\|00\rangle - \|11\rangle$	$\|10\rangle$
1 1	$\frac{1}{\sqrt{2}}\|10\rangle - \|01\rangle$	$\|11\rangle$

4. Bob observes his qubits. The table shows that the bits b_1 and b_2 are recovered correctly.

Remark: Regarding Alice's action in the beginning, it is sufficient and necessary to force the qubits into orthonormal states.

4 Quantum Algorithms

In this section we will finally introduce quantum algorithms. But first let us discuss why quantum computing gives us an incredible extreme speed-up over classical computers. Given an n-bit classical computer, the number of states is just 2^n. The computer will be in one of the 2^n possible states. Given an n-qubit quantum system, the quantum computer can be in a superposition of all possible 2^n states. For example, if we have a 3 qubit system, the set of basis states will be $\{|000\rangle, |001\rangle, \ldots, |111\rangle\}$. So in total there are $2^3 = 8$ basis states. In the general form, an n-qubit quantum system will have 2^n many basis states and the general state of the system will look like

$$|\psi\rangle = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle$$

such that $\alpha_n \in \mathbb{C}$ and $\sum |\alpha_x|^2 = 1$. For the reader to get used to the notation, we encourage the reader to explicitly write the general state of an 3-qubit system.

As the number of bits increase, the number of states grow exponentially. Exponential functions grow extremely fast. For moderate values of n, 2^n is even larger than the number of atoms in the universe. What is the source of the computational power of quantum computers? Suppose we have a k-level quantum system having k many basis states, and suppose we also have an l-level system having l many basis states. The composite system, using tensor product, will need $k \cdot l$ parameters to be defined. Assume we bought 16MB and 32MB of classical memory for our computer. In total we have 48MB of memory. If this was a quantum system, the composite system's memory would not be the sum but the *product* of the memory of two systems. A qubit lies in the vector space $\mathbb{C}^2$. If we had an n-qubit system, we would have to take the tensor product of the vector spaces

$$\underbrace{\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{n \text{ times}} = \mathbb{C}^{2^n}.$$

So we have an exponentially large vector space. But can we manipulate the state of an n-qubit system and the amplitudes efficiently? Even if we have a superposition of states, we should manipulate these states in a clever way so that when we measure the qubits we get a desired outcome with a high probability. We will be starting with some initial qubits set to a fixed state and then apply a sequence of unitary transformations to qubits.

This sequence of unitary transformations will form a quantum algorithm. Defining a quantum algorithm is an art. This will be the aim of this chapter, to see how we can make use of the nature's most powerful computer to solve problems.

4.1 Deutsch's Algorithm

We now look at perhaps the simplest quantum algorithm which is known as *Deutsch's algorithm*. This algorithm is to solve a problem about finding the character of a function.

Definition 29. Let $f : \{0,1\} \to \{0,1\}$ be a function. If $f(0) \neq f(1)$, we say that f is *balanced*. If $f(0) = f(1)$, then f is *constant*.

Problem. Given a function $f : \{0,1\} \to \{0,1\}$ as a black-box (we can evaluate but cannot look inside the function), tell if the function is constant or balanced.

A classical computer can solve this problem by evaluating f on both inputs, and then compare them to see whether f is balanced or constant. Is there a better solution? With a classical algorithm, no. However with quantum computing, this is possible indeed. A quantum computer can be in a superposition of all states. We shall use this property to evaluate both inputs at one time.

Consider a function f

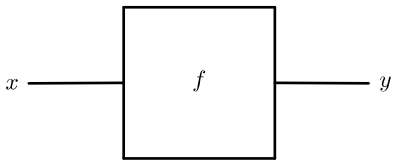


Figure 20: Classical representation of a function.

Such function could be thought of as a matrix. For example, the function f, where $f(0) = 0$, $f(1) = 0$ is represented as

$$f = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

Multiplying this with state $|0\rangle$ or with state $|1\rangle$ would result in state $|0\rangle$. For a quantum system we need the transformation to be unitary (hence reversible). If f is the name of the function, then the following black-box (oracle) U_f will be the quantum counterpart that we shall employ to evaluate input.

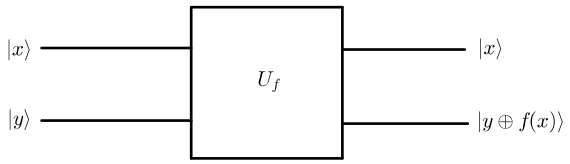


Figure 21: Quantum counterpart of the function f as an oracle.

The top input $|x\rangle$ will be the qubit we want to evaluate and $|y\rangle$ controls the input. The function U_f takes the state $|x, y\rangle$ to state $|x, y \oplus f(x)\rangle$. If $y = 0$, this simplifies to

$$|x, y\rangle = |x, 0 \oplus f(x)\rangle = |x, f(x)\rangle.$$

Clearly U_f is reversible. That is, if we give $|x, y \oplus f(x)\rangle$ as an input to U_f it will output $|x, y\rangle$. State $|x, y\rangle$ goes to $|x, y \oplus f(x)\rangle$ which further goes to U_f once more. That is,

$$U_f|x, y \oplus f(x)\rangle = |x, (y \oplus f(x)) \oplus f(x)\rangle = |x, y \oplus 0\rangle = |x, y\rangle$$

For the function f such that $f(0) = 1$ and $f(1) = 0$, the corresponding U_f is

$$U_f = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Each column represents the output when given the input $|x, y\rangle$.

So we are given some U_f as an oracle and we want to determine whether the function expressed by U_f is constant or balanced.

Notation. We shall denote a measurement by the "meter" figure

Figure 22: Measurement symbol.

First attempt:

Rather than evaluating f twice, we try to put the states into superposition. Instead of having the top input to be either in state $|0\rangle$ or $|1\rangle$, we put the top input in state

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

So we are applying the Hadamard matrix H on state $|0\rangle$.

$$H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Although it might not be correct, let us try to put the second qubit to state $|0\rangle$.

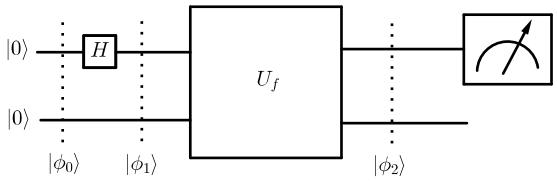


Figure 23: First attempt for Deutsch's problem.

So we have $U_f(H|0\rangle \otimes |0\rangle)$.
Let us examine each state.
$|\phi_0\rangle = |00\rangle$.
$|\phi_1\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}}$

After multiplying with U_f, we have

$$|\phi_2\rangle = \frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}$$

For the function f such that $f(0) = 1$ and $f(1) = 0$, the state $|\phi_2\rangle$ would be

$$|\phi_2\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}.$$

If we measure the first qubit we get $\frac{1}{2}$ chance of finding it in state $|0\rangle$ and $\frac{1}{2}$ chance in $|1\rangle$. Similarly for the second qubit. So this solution gives us no information.

Second attempt.

Rather than leaving the second qubit in state $|0\rangle$, let us put it in the superposition state

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

This is still a superposition of states but notice the phase change. We can obtain this state by multiplying the Hadamard matrix with the state $|1\rangle$. Let us leave the first qubit as an ambiguous $|x\rangle$.

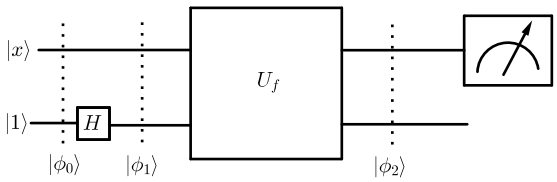


Figure 24: Second attempt for Deutsch's problem.

So here we have $U_f(x, H|1\rangle)$. Then the states are as follows:

$$|\phi_0\rangle = |x, 1\rangle$$

$$|\phi_1\rangle = |x\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \frac{|x0\rangle - |x1\rangle}{\sqrt{2}}$$

Applying U_f, we get

$$|\phi_2\rangle = \frac{|x, 0 \oplus f(x)\rangle - |x, 1 \oplus f(x)\rangle}{\sqrt{2}} = |x\rangle \left(\frac{|f(x)\rangle - |\overline{f(x)}\rangle}{\sqrt{2}} \right),$$

where $\overline{f(x)}$ denotes the complement of $f(x)$.
So we have

$$|\phi_2\rangle = \begin{cases} |x\rangle \left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}} \right), & \text{if } f(x) = 0 \\ \\ |x\rangle \left(\dfrac{|1\rangle - |0\rangle}{\sqrt{2}} \right), & \text{if } f(x) = 1 \end{cases}$$

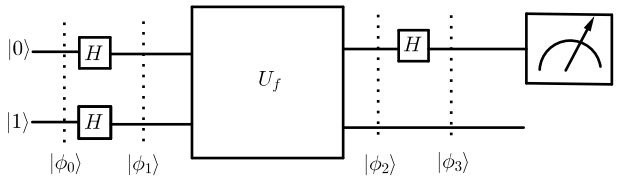
Recall that $a - b = (-1)(b - a)$. So we might write $|\phi_2\rangle$ as

$$|\phi_2\rangle = (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

What would happen if we measure either the first or the second state? Again, this does not give any information. The first qubit will be in state $|x\rangle$ and the second qubit will be either in state $|0\rangle$ or $|1\rangle$, with equal probability. We need something more.

Solution.

We put both first and second qubits into superposition. We shall also put the outcome of the first qubit through a Hadamard transformation, causing an interference.



Let us examine the states.

$|\phi_0\rangle = |01\rangle$

$$|\phi_1\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \frac{|00\rangle - |01\rangle + |10\rangle - |11\rangle}{2} = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}$$

We saw from the last attempt at solving this problem, that when we put the second qubit into superposition and then multiply by U_f we will be in a superposition

$$(-1)^{f(x)}|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}.\right)$$

Note that $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ is the eigenvector of U_f with the eigenvalue $(-1)^{f(x)}$. Now with $|x\rangle$ in a superposition, we have

$$|\phi_2\rangle = \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Let us carefully look at

$$(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle$$

If f is constant, this becomes either

$$+1(|0\rangle + |1\rangle) \text{ or } -1(|0\rangle + |1\rangle)$$

If f is balanced, it becomes either

$$+1(|0\rangle - |1\rangle) \text{ or } -1(|0\rangle - |1\rangle)$$

Summing up, we have that

$$|\phi_2\rangle = \begin{cases} (\pm 1)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right), & \text{if } f \text{ is constant} \\ (\pm 1)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right), & \text{if } f \text{ is balanced} \end{cases}$$

Remember that the Hadamard matrix is its own inverse and takes $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$ to $|0\rangle$ and takes $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ to $|1\rangle$. We apply the Hadamard matrix to the first qubit to get

$$|\phi_3\rangle = \begin{cases} (\pm 1)|0\rangle \left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}} \right), & \text{if } f \text{ is constant} \\ \\ (\pm 1)|1\rangle \left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}} \right), & \text{if } f \text{ is balanced} \end{cases}$$

Now we simply measure the first qubit. If it is in state $|0\rangle$, then we know that f is constant. Otherwise it is balanced. We did this with just one evaluation!

4.2 Deutsch-Jozsa Algorithm

We now give a straight forward generalization of the Deutsch's Problem. Instead of the single bit domain function, consider the function

$$f : \{0,1\}^n \to \{0,1\}$$

given as a black-box. We assume that f is either constant ($f(x)$ is the same for all x) or balanced ($f(x) = 0$ for exactly half of the input strings x, and $f(x) = 1$ for the other half of the inputs). We are asking the same question as before, whether f is constant or balanced.

Classically, the solution requires $2^{n-1} + 1$ queries in the worst case scenario. The quantum algorithm we present here will solve the problem by making only one query!

Similar to what we did in the Deutsch's algorithm, we define the unitary quantum oracle

$$U_f : |\mathbf{x}\rangle|y\rangle \to |\mathbf{x}\rangle|y \oplus f(\mathbf{x})\rangle$$

We are writing $\mathbf{x}$ in boldface since it refers to an n-bit string. $U_{f(\mathbf{x})}$ is now controlled by the register of n qubits in the state $|\mathbf{x}\rangle$. It is easy again to see that $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ is an eigenvector of $U_{f(\mathbf{x})}$ with the eigenvalue $(-1)^{f(\mathbf{x})}$.

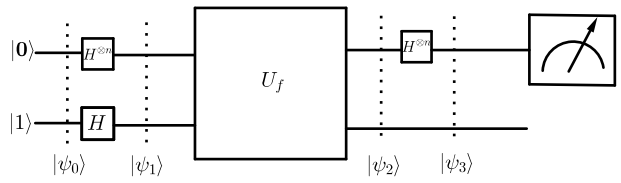


Figure 25: Circuit diagram for Deutsch-Jozsa algorithm.

Notice that instead of using 1 qubit Hadamard gate, we are using a Hadamard gate $H^{\otimes n}$ of n qubits. That is,

$$H^{\otimes n} = \underbrace{H \otimes \cdots \otimes H}_{n \text{ times}}.$$

We shall use $|0\rangle^{\otimes n}$, or $|\mathbf{0}\rangle$ to denote the state that is the tensor product of n qubits, each in state $|0\rangle$.

First step is to initialize the control register to prepare in state

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle.$$

Consider the action of an n-qubit Hadamard transformation on the state $|0\rangle^{\otimes n}$.

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots (|0\rangle + |1\rangle). \quad (n \text{ times})$$

Expanding the tensor product, this can be written as

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle.$$

So the state immediately after the first Hadamard transformation is

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right].$$

Note that the query register is now in an equally weighted superposition of all the possible n-bit input strings. Next we have the state after $U_{f(\mathbf{x})}$ gate.

$$\begin{aligned}|\psi_2\rangle &= \frac{1}{\sqrt{2^n}} U_f \left(\sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \right) \\ &= \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})} |\mathbf{x}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]\end{aligned}$$

Consider the action of the n-qubit Hadamard gate on an n-qubit basis state $|\mathbf{x}\rangle$. But first let us note that it is easy to see that the effect of the 1-qubit Hadamard gate on a 1-qubit basis state $|x\rangle$ can be written as

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

$$= \frac{1}{\sqrt{2}} \sum_{z \in \{0,1\}} (-1)^{xz} |z\rangle.$$

We can see that the action of the Hadamard transformation on an n-qubit basis state $|\mathbf{x}\rangle = |x_1\rangle|x_2\rangle \ldots |x_n\rangle$ is given by

$$H^{\otimes n}|\mathbf{x}\rangle = H^{\otimes n}(|x_1\rangle|x_2\rangle \ldots |x_n\rangle)$$

$$= H|x_1\rangle H|x_2\rangle \ldots H|x_n\rangle$$

$$= \tfrac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_1}|1\rangle)\tfrac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_2}|1\rangle) \cdots \tfrac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_n}|1\rangle)$$

$$= \frac{1}{\sqrt{2^n}} \sum_{z_1 z_2 \ldots z_n \in \{0,1\}^n} (-1)^{x_1 z_1 + x_2 z_2 + \cdots x_n z_n} |z_1\rangle|z_2\rangle \cdots |z_n\rangle.$$

The above equation can be then written as

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

where $\mathbf{x} \cdot \mathbf{z}$ denotes the bitwise inner product of $\mathbf{x}$ and $\mathbf{z}$, modulo 2.[4] We shall use this equation often, so it is important for the reader to keep this convention in mind. The state after the final n-qubit Hadamard transformation is

$$|\psi_3\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z} \in \{0,1\}^n} (-1)^{\mathbf{x} \cdot \mathbf{z}} |\mathbf{z}\rangle \right) \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$$

[4]The set of all binary strings of length n is defined on the space $\{0,1\}^n$. This, however, is not interpreted as a real vector space. The space of bit strings is vector space over a finite field (in particular, it is a field of characteristic 2). In such vector fields, there are many "intuitions" (such as orthogonality, projection length etc.) from vector fields over $\mathbb{R}$ or $\mathbb{C}$ that no longer apply. When u and v are two binary strings of length n, their bitwise inner product is also called a *parity check* or *checksum* since it indicates whether the second vector 'satisfies' the parity check specified by the first vector (or equivalently whether the first vector satisfies the parity check specified by the second). To satisfy a parity check u, a vector v must have an even number of 1's at the positions (coordinates) specified by the 1's in u. So if $u \cdot v = 0$, this is not interpreted as u and v are orthogonal like in real vector spaces, but it means that it satisfies parity check and so v has an even number of 1's at the positions specified by the 1's in u (or also vice versa).

$$= \frac{1}{2^n} \sum_{\mathbf{z} \in \{0,1\}^n} \left(\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x}) + \mathbf{x} \cdot \mathbf{z}} \right) |\mathbf{z}\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right].$$

At the end of the algorithm a measurement of the first register is made in the computational basis. To see what happens, consider the total amplitude of $|\mathbf{z}\rangle = |0\rangle^{\otimes n}$ in the first register of state $|\psi_3\rangle$. This amplitude is

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

Consider this amplitude in the two cases.

Case 1: f is constant. If f is constant, the amplitude of $|0\rangle^{\otimes n}$ is either 1 or -1, depending on which value $f(\mathbf{x})$ takes. So if f is constant, a measurement of the first register is certain to return $|0\rangle^{\otimes n}$.

Case 2: f is balanced. If f is balanced, then it is easy to see that the positive and negative amplitudes interfere (cancel) each other since half of the amplitudes would be positive and the other half would be negative. In this case, the amplitude of $|0\rangle^{\otimes n}$ is 0. So if f is balanced, a measurement of the first register is certain not to return $|0\rangle^{\otimes n}$.

To determine whether f is balanced or constant, it is sufficient to measure the first register. If the result is $|0\rangle^{\otimes n}$, then f is constant. Otherwise f is balanced. Again, notice that we determined the characteristic of f by making a single query by giving the register just the state $|0\rangle^{\otimes n}$.

4.3 Simon's Algorithm

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$ such that $f(x) = f(y)$ if and only if $x = y$ or $y = x \oplus s$ for some hidden string $s \in \{0,1\}^n$.

In other words, the values of f repeat themselves in some pattern and the pattern is determined by s. We shall call s the *period* of f. *Simon's Problem* is to find the period s.

For example, if $n = 3$, then the following function is an example of a function that satisfies the required property:

x	$f(x)$
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

In this instance, the string s is 110. Every output of f occurs twice, and the two input strings corresponding to any one given output have bitwise XOR equal to $s = 110$.

Note that if $s = 0^n$, then the function must be one-to-one. Otherwise the function is two-to-one.

How can we solve this problem using a classical algorithm? We would have to evaluate f on different binary strings. After each evaluation, we check to see if that output has already been found. If we find two inputs x_1 and x_2 such that $f(x_1) = f(x_2)$, then we are assured that

$$x_1 = x_2 \oplus s$$

and we can obtain s by $\oplus$-ing both sides with x_2:

$$x_1 \oplus x_2 = x_2 \oplus s \oplus x_2 = s$$

If the function is a two-to-one function, then we will not have to evaluate more than half the inputs before we get a repeat. If we evaluate more than half the strings and still cannot find a match, then we know that f is one-to-one and that $s = 0^n$. Hence, in the worst case scenario, we need $2^{n-1} + 1$ evaluations.

The quantum part of Simon's algorithm is to evaluate the following several times:

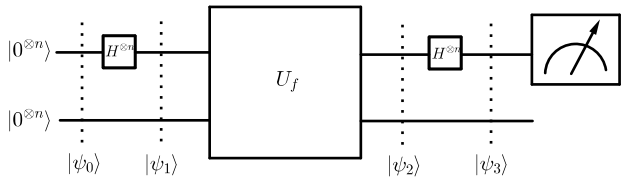


Figure 26: Quantum part of the Simon's algorithm.

Let us look at each stage. It is clear that

$$|\psi_0\rangle = |\mathbf{0}, \mathbf{0}\rangle.$$

We then place the first register in a superposition of all possible strings of $\{0,1\}^n$. From the previous examples, we know that

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}, \mathbf{0}\rangle.$$

Evaluation of U_f on gives us

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}, f(\mathbf{x})\rangle.$$

Finally we apply $H^{\otimes n}$ to the first register and get

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{\mathbf{x}\in\{0,1\}^n} \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{x}\cdot\mathbf{z}} |\mathbf{z}, f(\mathbf{x})\rangle.$$

Note that there are exactly two $\mathbf{x}$'s, call them $\mathbf{x}_1$ and $\mathbf{x}_2$, for which $f(\mathbf{x}_1) = f(\mathbf{x}_2)$. Since $\mathbf{x}_2 = \mathbf{x}_1 \oplus \mathbf{s}$, we can rewrite $|\psi_3\rangle$ as

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{\mathbf{x}_1,\mathbf{x}_2\in\{0,1\}^n} \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{z}\cdot\mathbf{x_1}} + (-1)^{\mathbf{z}\cdot\mathbf{x_2}} |\mathbf{z}, f(\mathbf{x}_1)\rangle,$$

where $\mathbf{x}_2 = \mathbf{x}_1 \oplus \mathbf{s}$. Let us look at the coefficient of $|\mathbf{z}, f(\mathbf{x}_1)\rangle$.

The coefficient of $|\mathbf{z}, f(\mathbf{x}_1)\rangle$ is 0 when $(-1)^{\mathbf{z}\cdot\mathbf{x_1}} + (-1)^{\mathbf{z}\cdot\mathbf{x_2}}$ is 0. Given that $\mathbf{x}_2 = \mathbf{x}_1 \oplus \mathbf{s}$, this means that the coefficient of $|\mathbf{z}, f(\mathbf{x}_1)\rangle$ is 0 when

$(-1)^{\mathbf{z}\cdot\mathbf{x_1}} + (-1)^{\mathbf{z}\cdot\mathbf{x_1}\oplus\mathbf{s}} = (-1)^{\mathbf{z}\cdot\mathbf{x_1}} + (-1)^{\mathbf{z}\cdot\mathbf{x_1}\oplus\mathbf{z}\cdot\mathbf{s}} = (-1)^{\mathbf{z}\cdot\mathbf{x_1}} + (-1)^{\mathbf{z}\cdot\mathbf{x_1}}(-1)^{\mathbf{z}\cdot\mathbf{s}}$ is 0.

So when $\mathbf{z} \cdot \mathbf{s} = 1$, the coefficient of $|\mathbf{z}, f(\mathbf{x}_1)\rangle$ is 0. Hence, upon measuring the first register, we will only find those $\mathbf{z}$ which are orthogonal to $\mathbf{s}$, i.e., those which satisfy $\mathbf{z} \cdot \mathbf{s} = 0$.

This determines a subspace which $\mathbf{s}$ must lie on. Given n many samples of $\mathbf{z}$, $\mathbf{s}$ is constrained to a 0-dimensional subspace and we can solve the n linear equations using classical methods, Gaussian elimination for example, to determine $\mathbf{s}$.

Simon's algorithm requires $O(n)$ queries to the black-box, whereas a classical algorithm would need at least $\Omega(2^{n/2})$ queries. We should also note that Simon's algorithm is optimal in the sense that any quantum algorithm to solve this problem requires $\Omega(n)$ queries.

4.4 Grover's Search Algorithm

How do you find needle in a haystack? We have to look at each piece of hay separately and check each one to see if it is the desired needle. We may also convert this problem to finding a special element, or the index of the special entry, in an unstructured list or database as Lov Grover himself puts it.

Problem. Given a blackbox for computing the function $f : \{0,1\}^n \to \{0,1\}$ such that $f(x^*) = 1$; $f(x) = 0$ if $x \neq x^*$. Find the key value x^*.

Let us assume for convenience that there are $N = 2^n$ elements to be looked at. Classically, we are required to check $N - 1$ elements, provided that the element is in the list. We do not need to check the last element as it will be the desired entry. So that means, using classical algorithm, we need $O(N)$ steps to find the key value. Of course if the list were ordered, say in an increasing order, we could apply *binary search* for instance and obtain a solution with logarithmic complexity. With quantum computing the unstructured search can be done in $O(\sqrt{N})$ steps using *Grover's algorithm*. So we have a quadratic speed-up over the classical algorithm. Let us assume that, for convenience, there is a unique x^* such that $f(x^*) = 1$. So by definition, $f(x) = 1$ if x is the key value (i.e. the solution), and $f(x) = 0$ if x is not a solution to the search problem. Suppose also that we are provided with an *oracle*, whose internal workings is not important for us now, with the ability to *recognize* solution to the search problem.

The discussion of the oracle without seeing how it works in practice is rather abstract and maybe even puzzling. It seems as if the solution is kept in a register in the oracle and it *knows* the answer. But then why bother with a search algorithm if there is an oracle already to consult? There is

a difference between *knowing* the solution and being able to *recognize* the solution when given. A simple example to this is the distinction between proof checking and finding a proof. The former problem is fairly easy as the proof is already given and all we need to do is to check if the proof is logically correct. The latter problem is actually very hard, and in fact, unsolvable in some cases. Finding a proof of a sentence, say in first-order logic, is known to be a hard problem. The former problem is Turing-computable, the latter is Turing-recognizable.

The quantum oracle, similar to that in the previous algorithms, is a unitary operator O defined as

$$O : |x\rangle|q\rangle \rightarrow |x\rangle|q \oplus f(x)\rangle,$$

where $|x\rangle$ is the index register, such that $x \in \{0,1\}^n$, $\oplus$ denotes addition modulo 2 (i.e. bitwise XOR), and $|q\rangle$ is the oracle single qubit.

First let us try solving this problem by placing $|x\rangle$ into an equally weighted superposition of all possible strings and then evaluating the oracle O on state

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle.$$

After applying the oracle, we get the state

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x, f(x)\rangle.$$

Measuring the first register will give one of the N strings with equal probability. Measuring the second register will give $|0\rangle$ with probability $\frac{N-1}{N}$, and give $|1\rangle$ with probability $\frac{1}{N}$ since there is only one solution x such that $f(x) = 1$ by definition of the problem. If we get lucky enough to measure $|1\rangle$ on the second qubit, then the first qubit will have the correct answer since the first and second registers are entangled. However, it is highly improbable that we get so lucky. So we need something more innovative.

Grover's algorithm uses two tricks. First trick is called the *phase inversion*, which "marks" the key value by changing its phase. It works as follows: We set the oracle qubit in the state $\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}$ which is basically equal to $H|1\rangle$. The action of the oracle is thus

$$O : |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \rightarrow (-1)^{f(x)}|x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

Now notice that $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ is an eigenvector, so we can omit this qubit and the action of the oracle may be written as

$$O : |x\rangle \rightarrow (-1)^{f(x)}|x\rangle.$$

Now if we look carefully, the oracle *marks* the key value by inverting its phase.

So how does the phase inversion act on states? First let us give a small illustration of what we mean by that. If $|x\rangle$ starts in an equal superposition of four different states, i.e. $[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]^T$, and assume that the key value is the computational basis state 10. After performing the phase change, the state will become $[\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}]^T$. Measuring $|x\rangle$ does not give enough information unfortunately. All states have equal probabilities. Changing the phase has no effect on the outcome. What we need is *amplifying* the amplitude of the key value.

The second trick of Grover's algorithm is called the *inversion about the mean*. As an example, consider a sequence of integers: $53, 38, 17, 23, 79$. The average of these numbers is $\mu = 42$. If we want to invert an element α around the mean μ, we get $\alpha' = \mu + (\mu - \alpha)$ or simply

$$\alpha' = 2\mu - \alpha.$$

So given a general state

$$\sum_k \alpha_k |k\rangle,$$

the inversion about the mean will produce the state

$$\sum_k (2\mu - \alpha_k)|k\rangle,$$

where $\mu = \frac{1}{N}\sum_k \alpha_k$.

How do we define the inversion about the mean in terms of matrices then? Let us again first try to describe it by giving an example and then later define it more precisely. Consider the vector $V = [53, 38, 17, 23, 79]^T$.

Consider the matrix

$$A = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

It is easy to see that A is a matrix that finds the average of the elements of the vector V.

$$AV = [42, 42, 42, 42, 42]^T$$

In terms of matrices, the formula $\alpha' = 2\mu - \alpha$ becomes

$$V' = 2AV - V = (2A - I)V.$$

In general, if we were to deal with 2^n possible states, then

$$A = \begin{bmatrix} \frac{1}{2^n} & \frac{1}{2^n} & \cdots & \frac{1}{2^n} \\ \frac{1}{2^n} & \frac{1}{2^n} & \cdots & \frac{1}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2^n} & \frac{1}{2^n} & \cdots & \frac{1}{2^n} \end{bmatrix}$$

So the operator $(2A - I)$ inverts the given vector about its mean. But how do we actually implement this? We will need to define an n-qubit operator $U_0^{\perp}$ acting as follows

$$U_0^{\perp} = \begin{cases} |x\rangle \to -|x\rangle, & x \neq 0 \\ |0\rangle \to |0\rangle \end{cases}$$

This operator applies a phase inversion to all n-qubit states that are orthogonal to the state $|0\rangle$. In terms of matrices

$$U_0^{\perp} = (2|0\rangle\langle 0| - I) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix}$$

This operator inverts the phase of the states that are orthogonal to the state $|0\rangle$. But we are not done quite yet. We want to invert the phase of the states that are orthogonal to the general superposition of states.

Let us suppose that

$$|\psi\rangle = H|0\rangle = \frac{1}{\sqrt{N}} \sum_k |k\rangle.$$

What we can do is the following. We can first transform $|\psi\rangle$ into $|0 \dots 0\rangle$. Then we can do inversion about $|0 \dots 0\rangle$, and then we can transform $|0 \dots 0\rangle$ back into $|\psi\rangle$. For this, consider the action of the operator

$$HU_0^{\perp}H.$$

Now it is easy to observe that $HU_0^{\perp}H = (2|\psi\rangle\langle\psi| - I)$.[5] To see why this operator inverts about the mean, note that

$$HU_0^{\perp}H = H \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix} H$$

$$= H \left(\begin{bmatrix} 2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} - I \right) H$$

$$= H \begin{bmatrix} 2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} H - HIH$$

[5] By H, we of course mean $H^{\otimes n}$.

$$= \begin{bmatrix} \frac{2}{\sqrt{N}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{\sqrt{N}} & 0 & \cdots & 0 \end{bmatrix} H - I \text{ (since } HH = I)$$

$$= \begin{bmatrix} \frac{2}{N} & \cdots & \frac{2}{N} \\ \vdots & \ddots & \vdots \\ \frac{2}{N} & \cdots & \frac{2}{N} \end{bmatrix} - I = \begin{bmatrix} \frac{2}{N} - 1 & & & \\ & \ddots & 2/\mathrm{N} & \\ & 2/\mathrm{N} & \ddots & \\ & & & \frac{2}{N} - 1 \end{bmatrix}.$$

Now why does this matrix invert a given vector about the mean? Let us take a look at what happens at the xth entry of the vector when given a state.

$$\begin{bmatrix} \frac{2}{N} - 1 & & & \\ & \ddots & 2/\mathrm{N} & \\ & & \ddots & \\ & 2/\mathrm{N} & & \frac{2}{N} - 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_x \\ \vdots \\ \alpha_{N-1} \end{bmatrix} = \begin{bmatrix} \vdots \\ \vdots \\ \frac{2}{N} \sum_{y=0}^{N-1} \alpha_y - \alpha_x \\ \vdots \\ \vdots \end{bmatrix}$$

Since $\frac{2}{N} \sum_y \alpha_y = 2\mu$, the xth entry of the resultant vector becomes $2\mu - \alpha_x$ which is exactly what we wanted to do. In other words, $HU_0^{\perp}H$ inverts the phase of the states that are orthogonal to $|\psi\rangle$. The set of states orthogonal to $|\psi\rangle$ is spanned by the states $H|x\rangle$ for $x \neq 0$. So for all $x \neq 0$,

$$(HU_0^{\perp}H)H|x\rangle = -H|x\rangle.$$

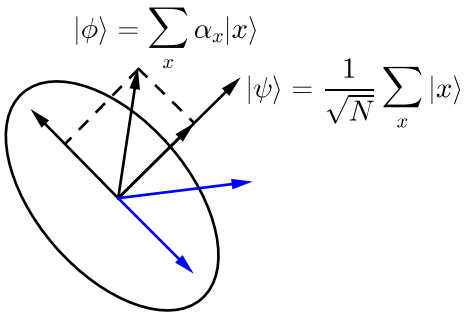


Figure 27: Reflection around $|\psi\rangle$ is actually a reflection around the mean. Given $|\phi\rangle$ to be inverted, we take the subspace of vectors that are orthogonal to $|\psi\rangle$, we take the orthogonal component of $|\phi\rangle$, invert it and we simply take the reflection of $|\phi\rangle$ in the direction of this inverted orthogonal component.

Now what happens if we use the phase inversion operator O and the inversion about the mean together? Suppose that we prepared our qubits into an equal superposition $\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$.

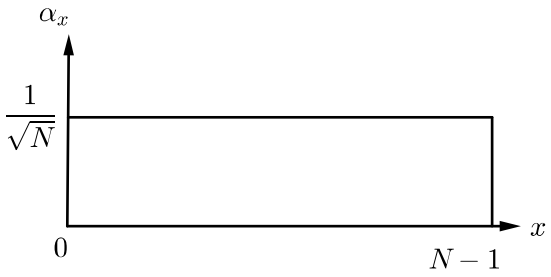


Figure 28: Putting states into an equal superposition.

If x^* is the key value we are looking for, the phase inversion operator inverts the phase of $|x^*\rangle$.

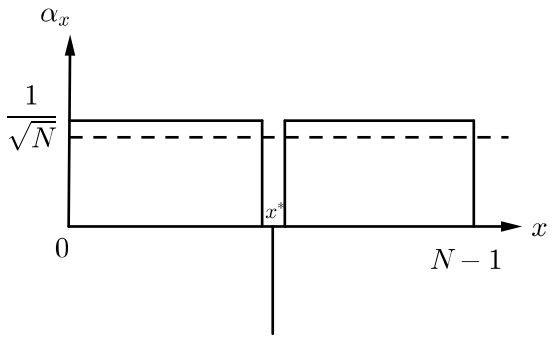


Figure 29: Phase inversion of the target value.

Since we invert the phase of $|x^*\rangle$, the average mean now drops down a little bit. What happens when we invert the phase of all the states around the mean? What happens is that the amplitude of $|x^*\rangle$ now goes up approximately to $\frac{3}{\sqrt{N}}$, while the amplitude of the non-key values goes down.

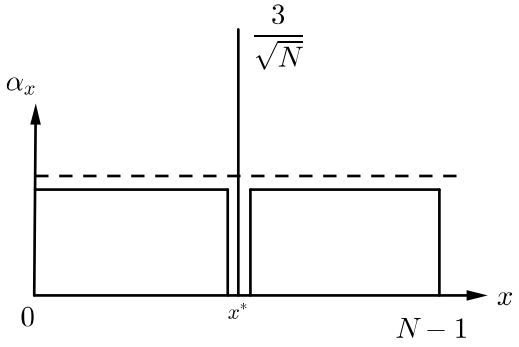


Figure 30: Inversion about the mean will amplify the the amplitude of the target value.

So if we iterate this process $O(\sqrt{N})$ times we get a sufficiently high probability of getting $|x^*\rangle$ when measuring the qubits. We call the iteration $HU_0^{\perp}HO$, *Grover's iteration.*

Hence, *Grover's algorithm* can be stated as follows.

Step 1. Start with a state $|\mathbf{0}\rangle$.
Step 2. Apply $H^{\otimes n}$.
Step 3. Repeat Grover's iteration $O(\sqrt{N})$ times. That is,

1. Apply the phase inversion operator O.
2. Apply the inversion about the mean operator $HU_0^{\perp}H$.

Step 4. Measure the qubits.

We might view this algorithm as follows.

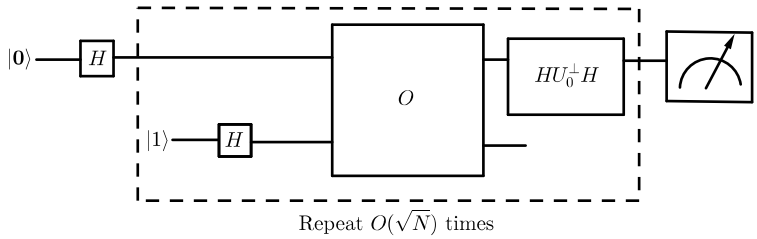


Figure 31: Diagram of Grover's algorithm.

There is a beautiful geometric interpretation of Grover's algorithm. In fact, Grover's iteration can be seen as a rotation in the two-dimensional space spanned by the starting vector $|\psi\rangle$ and the desired state $|x^*\rangle$. Actually, $|\psi\rangle$ and $|x^*\rangle$ are nearly orthogonal to each other for large N, but never entirely orthogonal since the inner product of $|\psi\rangle$ and $|x^*\rangle$ is $\frac{1}{\sqrt{N}}$. Let us define a state that is orthogonal to $|x^*\rangle$. Call this state $|x_{\perp}^*\rangle$.

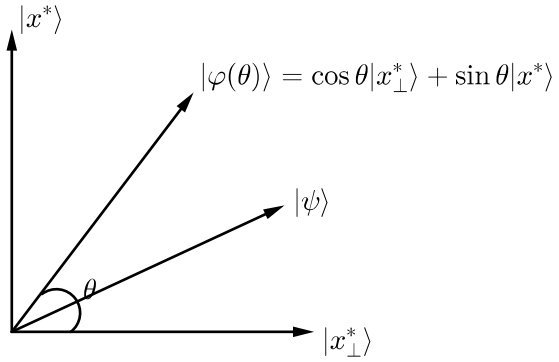


Figure 32: $|x^*\rangle$ denotes the solution space, $|x^*_\perp\rangle$ denotes the non-solution space, $|\psi\rangle$ is the initial equal superposition of states and $|\varphi(\theta)\rangle$ is the general trigonometric representation of any vectors lying on the plane spanned by $|x^*\rangle$ and $|x^*_\perp\rangle$.

Consider now a family of states all lying in this plane spanned by $|x^*\rangle$ and $|x^*_\perp\rangle$. The general form of a vector in this space has the form $\cos\theta|x^*_\perp\rangle + \sin\theta|x^*\rangle$, where θ is a real parameter. The desired state, namely $|x^*\rangle$, is in this parameterized family with $\theta = \frac{\pi}{2}$.

Before we start the Grover's iteration, we are in state $|\psi\rangle$. The state can also be written in the general trigonometric form with $\theta = \arcsin\frac{1}{\sqrt{N}}$. So what is the effect of one Grover iteration on $|\varphi(\theta)\rangle$? We begin with inverting the phase of the key state $|x^*\rangle$ which is just a reflection around $|x^*_\perp\rangle$ where we change the sign of the coefficient of $|x^*\rangle$. That is, $O|\varphi(\theta)\rangle = \cos\theta|x^*_\perp\rangle - \sin\theta|x^*\rangle$.

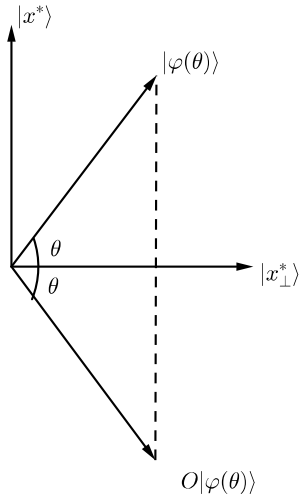


Figure 33: Geometric representation of phase inversion step.

Next we do $HU_0^\perp H$. As we noted earlier, this is a reflection around the state $|\psi\rangle$. The Grover iteration $HU_0^\perp HO$ is a product of two reflections which is a rotation twice the angle between $|\psi\rangle$ and $|x^*_\perp\rangle$, towards $|x^*\rangle$.

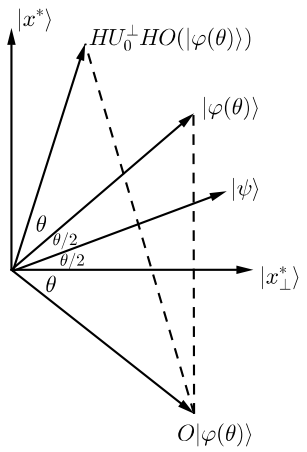


Figure 34: Inversion about the mean is just a reflection about $|\psi\rangle$. Combining phase inversion and the inversion about the mean operations, we end up with a rotation towards the target vector.

So each iteration rotates the state a little closer to the key state $|x^*\rangle$. But we have to be careful that if we over iterate the process, we pass the key state and get further away. So it is crucial to choose the right number of iteration, which is approximately $\sqrt{N}$. For N item search problem with M many key values, it turns out that we only need to apply the oracle $O\left(\sqrt{\frac{N}{M}}\right)$ times in order to obtain a solution on a quantum computer.

We should also note that the state does not hit the key state exactly but gets very near. Except for the special case when $N = 4$, where a single oracle call is enough to do the search with a correct answer with probability exactly 1. However, for the other cases, the output is a state that is not equal to but very close to the key state.

4.5 Shor's Factoring Algorithm

Given two integers, it is an easy task to multiply them together. It is however very hard in general to perform the inverse. That is, it is a harder problem to factorize a given integer N. The *Fundamental Theorem of Arithmetic* tells us that any natural number greater than 1 is either prime itself or is the product of prime numbers. Moreover, this product is unique. In this section, we introduce a polynomial time quantum algorithm for integer factorization.

There is an efficient algorithm, called Euclid's algorithm, for determining the greatest common divisor of two given numbers a and b. Given two natural numbers a and b, the function $gcd(a, b)$ does the following until $a = b$: If $a > b$ then define the new value of a to be $a - b$, otherwise define the new value of b to be $b - a$. If $a = b$, then stop and define a to be the output of $gcd(a, b)$. The recursive definition of $gcd(a, b)$ is as follows.

$$gcd(a,b) = \begin{cases} a & \text{if } a = b \\ gcd(a-b,b) & \text{if } a > b \\ gcd(a,b-a) & \text{if } a < b \end{cases}$$

Integer factorization problem is to find the prime factorizations of a given integer N. Prime factorization of integers is commonly used in RSA cryptography. The reason why RSA cryptography systems use integer factorization is because there is no efficient classical algorithm known yet which factorizes large numbers. For the worst case scenario, let $N = pq$ for two large prime numbers. Given N, if we want to find its prime factors p and q, we can do exhaustive search in the worse case. Since the input size is

measured with bits, the size of N is approximately $w = \log_2 N$, i.e. size of the binary representation of N. Therefore, $O(N) = O(2^{\log_2 N}) = O(2^w)$. So the exhaustive search takes exponential time in the input size.

The problem of factorizing integers can be converted into the problem of period finding using modular arithmetic. We say that $a \equiv x \mod N$ if x is the remainder of $\frac{a}{N}$. The famous mathematician Leonard Euler discovered a very useful relationship between exponential functions and modular arithmetic. Let us examine the function $f(k) = 3^k$.

$3^1 = 3$
$3^2 = 9$
$3^3 = 27$
$3^4 = 81$
$3^5 = 243$
$3^6 = 729$
$3^7 = 2187$
$3^8 = 6561$

Mod10 of this function gives us

$3^1 \mod 10 = 3$
$3^2 \mod 10 = 9$
$3^3 \mod 10 = 7$
$3^4 \mod 10 = 1$
$3^5 \mod 10 = 3$
$3^6 \mod 10 = 9$
$3^7 \mod 10 = 7$
$3^8 \mod 10 = 1$

What we notice is a repeating sequence

$$3, 9, 7, 1, 3, 9, 7, 1 \ldots$$

Moreover the last element of the cycle is always 1. For a given function

$$f_{a,N}(x) = a^x \mod N,$$

as long as a and N are relatively prime, that is $gcd(a, N) = 1$, we have this cyclic property. The least number r such that $a^r \mod N = 1$ is called the *period* of $f_{a,N}(x)$. The period of $f_{3,10}(x)$ is for instance 4, as it can

be seen. Apparently integer factoriztion can be reduced to the problem of period finding.

The algorithm for integer factorization is as follows. Suppose that we are given $N = pq$ for two prime numbers p and q. The aim is to find p and q.

Step 1. Pick any integer $a < N$. See if a and N are relatively prime. If $gcd(a, N) \neq 1$, then the common factor must be a factor of N, so we found one of p or q. In this case we can output the factors and we are done.

If $gcd(a, N) = 1$, continue with Step 2.

Step 2. Compute the period of $f_{a,N}(x) = a^x \mod N$ (This is a hard problem and it takes exponential time using classical algorithms). Call this period r. See if r is even. If not, go to Step 1. If so, see if $a^{\frac{r}{2}} + 1 \equiv 0 \mod N$. If $a^{\frac{r}{2}} + 1 \equiv 0 \mod N$, then go to Step 1. If $a^{\frac{r}{2}} + 1 \not\equiv 0 \mod N$, then continue with Step 3.
(Note that $f_{a,N}(r+s) = f_{a,N}(s)$ as the sequence will simply repeat after the first stage the function outputs 1.)

Step 3. Rearrange with algebra: We know that $a^r \equiv 1 \mod N$. So $a^r - 1 \equiv 0 \mod N$. Then, there must exist some k such that

$$a^r - 1 = k \cdot N$$

Since we know at this stage that r was an even number,

$$(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) = k \cdot N = k \cdot p \cdot q \qquad (*)$$

Step 4. Solve p and q as follows.

Let $p = gcd(a^{\frac{r}{2}} - 1, N)$ and let
$q = gcd(a^{\frac{r}{2}} + 1, N)$.

So why does this algorithm work? In the equation $(*)$, p must divide one of the factors on the left hand side of $(*)$ and q must divide the other factor. However, they cannot divide the same factor since that factor would be divisible by N. But neither factor is divisible by N since we assumed that $a^{\frac{r}{2}} + 1 \not\equiv 0 \mod N$ and we also assumed that r is the least number such that $a^r \equiv 1 \mod N$.

Then $(a^{\frac{r}{2}} - 1) \not\equiv 0 \mod N$. Since p and q divides separate factors on the left hand side of $(*)$, we can assume p divides $(a^{\frac{r}{2}} - 1)$, and q divides $(a^{\frac{r}{2}} + 1)$.

Example. Let $N = 35$. Find the prime factors.

Choose $a = 8$. Since $gcd(8, 35) = 1$, we continue with the next step. The period of $f_{a,N}(x)$ is 4 since we observe that

Powers of 8: $8, 64, 512, 4096, \ldots$
$8^x \mod 35$: $8, 29, 22, 1, \ldots$

Now $r = 4$ is an even number. Next is to check if $8^{\frac{4}{2}} + 1 \not\equiv 0 \mod 35$. This clearly holds.

Therefore, $(8^2 - 1)(8^2 + 1) = k \cdot 35$ for some k.

So we let $p = gcd(63, 35) = 7$, and let $q = gcd(65, 35) = 5$.

The problem here is that period finding is an extremely difficult task for classical computers. It turns out very easy for quantum computers though. It is not known whether there exists a polynomial time classical algorithm for finding the period. On the other hand, Shor's algorithm can compute this in polynomial time. This is an exponential speed-up over the best known classical algorithms.

Shor's algorithm is really *the* quantum algorithm that attracted many scientists. At the heart of integer factorization is period finding. At the heart of period finding is phase estimation, which relies on quantum Fourier transform (QFT).

Before explaning the quantum Fourier transform we shall review nth roots of unity.

Roots of Unity.

Recall that a complex number c can be given in polar form $c = (\rho, \theta)$, where ρ is the magnitude and θ is the phase. For unit length vectors, we simply take $\rho = 1$. But suppose that we are not working with unit length vectors and suppose we want to take the nth power of some complex number c. The nth power of c is just

$$c^n = (\rho^n, n\theta).$$

Raising to the nth power is basically multiplying n times. The figure below

shows a complex number and its first, second and third powers.

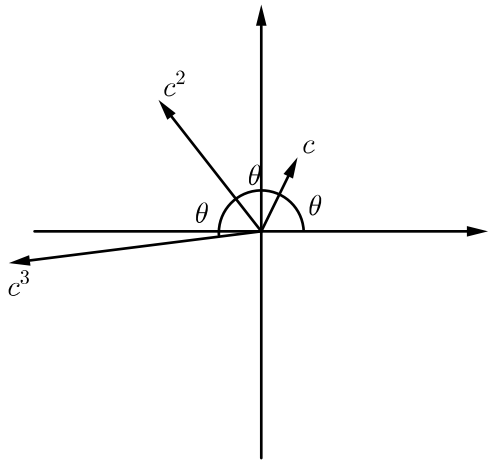


Figure 35: A complex number c and its square and cube.

We follow the same method in multiplying two complex numbers. If we have two complex numbers defined on the unit circle as

$x = \cos\theta_1 + i\sin\theta_1 = e^{i\theta_1}$ and
$y = \cos\theta_2 + i\sin\theta_2 = e^{i\theta_2}$,

Then $x \cdot y$ is obtained as follows.

$x \cdot y = (\cos\theta_1 + i\sin\theta_1)(\cos\theta_2 + i\sin\theta_2)$
$= \cos(\theta_1 + \theta_2) + i\sin(\theta_1 + \theta_2)$
$= e^{i(\theta_1+\theta_2)}$.

So multiplying complex numbers of unit length is just adding their phases. Let us now look at the roots of complex numbers. Given a complex number c in the polar form, its nth root is

$$c^{\frac{1}{n}} = \left(\rho^{\frac{1}{n}}, \frac{1}{n}\theta\right).$$

There is more to this however. Recall that the phase is defined only up to multiples of 2π. Therefore, we can rewrite the equation given above as

$$c^{\frac{1}{n}} = \left(\rho^{\frac{1}{n}}, \frac{1}{n}(\theta + 2k\pi)\right).$$

It appears that there are several nth roots of the same number. Considering the last equation, how many different solutions can we generate by varying k? In fact, there are exactly n many nth roots for a complex number.

$$\begin{array}{r|l} k = 0 & \frac{1}{n}\theta \\ k = 1 & \frac{1}{n}\theta + \frac{1}{n}2\pi \\ \vdots & \vdots \\ k = n-1 & \frac{1}{n}\theta + \frac{n-1}{n}2\pi \end{array}$$

When $k = n$, we obtain the first solution, when $k = n + 1$ we obtain the second solution. So we get a cyclic sequence. Let us assume that we are looking for the nth roots of unity. That is, we want to obtain all complex solutions to the equation

$$x^n = 1.$$

Again, it turns out that there are n many nth roots of unity. Setting $c = (1, 0)$ as its magnitude and phase ($\rho = 1, \theta = 0$), we have

$$c^{\frac{1}{n}} = (1,0)^{\frac{1}{n}} = \left(1^{\frac{1}{n}}, \frac{1}{n}(0 + k2\pi)\right) = \left(1, \frac{2k\pi}{n}\right).$$

By permitting $k = 0, 1, 2, \ldots, n-1$, we get exactly n different roots of unity. Note that when $k = n$, we get back to the first root. When $k = 1$, we obtain the *primitive nth root of unity*. The kth nth root of unity in exponential form is $e^{\frac{2\pi ik}{n}}$. We denote these n distinct roots of unity by

$$\omega^0 = 1, \omega^1, \omega^2, \ldots, \omega^{n-1}.$$

We show this by dividing a unit circle to n equal slices each of which is has an angle $\dfrac{2\pi}{n}$. For example, the 8th roots of unity can be viewed as in Figure 36.

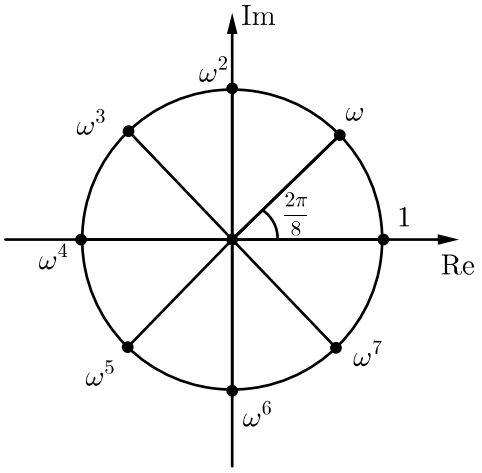


Figure 36: 8th roots of unity.

Note that

$$1 + \omega + \omega^2 + \ldots + \omega^{n-1} = 0.$$

since all vectors cancel each other. In general,

$$1 + \omega^j + \omega^{2j} + \ldots + \omega^{(n-1)j} = 0,$$

provided that $j \neq 0$, since the expression is equal to the geometric series

$$\frac{\omega^{nj} - 1}{\omega^j - 1}.$$

As long as $j \neq 0$, the denominator is non-zero. But then since ω^n is equal to 1, the numerator becomes 0 so the sum of the geometric series is equal to 0 when $j \neq 0$. If $j = 0$, then clearly the sum is equal to n.

Another important thing to note about roots of unity is their conjugate. Note that conjugate of ω^j is $(\omega^j)^* = \omega^{n-j}$. Since $\omega^n = 1$, we have that

$$\omega^{n-j} = \omega^n \cdot \omega^{-j} = 1 \cdot \omega^{-j} = \frac{1}{\omega^j}.$$

So in the exponential notation, the kth nth root of unity is $\omega^k = e^{\frac{2\pi ik}{n}}$ and its conjugate is $(\omega^k)^* = e^{-\frac{2\pi ik}{n}}$.

4.5.1 Quantum Fourier Transform

The Fourier transform in classical computation is one of the most practical tools in science and mathematics. It maps time dependent functions to their frequency domains. It treats periodic functions in such a way that a periodic function of period $r > 0$ is mapped to a function whose frequency amplitude is non-vanishing only at frequencies which are integer multiples of $1/r$. QFT is actually the quantum analogue of the discrete Fourier transform in classical computation. The discrete Fourier transform takes an N-dimensional vector of complex numbers $x_0, x_2, \ldots x_{N-1}$ and it outputs another N-dimensional complex vector $y_0, y_1, \ldots, y_{N-1}$ such that

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi i j k}{N}} x_j.$$

The quantum Fourier transform (QFT) looks very much like the discrete Fourier transform except that the notation is a little different. Suppose that we are given a set of basis states $\{|0\rangle, |1\rangle, \ldots |N-1\rangle\}$. QFT is a linear operator defined by the following action on a basis state

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |j\rangle. \qquad (*)$$

Equivalently, this transformation can be written as

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle,$$

where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j. We shall show shortly that QFT_N is unitary. But first let us show how we implement it using a quantum computer.

We shall assume that $N = 2^n$ for some natural number n. So the basis states are $|0\rangle, \ldots, |N-1\rangle$. We can represent any given integer j in binary represenataion $j = j_1 j_2 \ldots j_n$ such that $j = j_1 2^{n-1} + j_2 2^{n-2} + \ldots + j_n 2^0$. Binary fractions in the form $0.j_l j_{l+1} \ldots j_m$ represents

$$\frac{j_l}{2} + \frac{j_{l+1}}{2^2} + \ldots + \frac{j_m}{2^{m-l+1}}.$$

A very useful representation of the action of QFT_N is as follows.

$$\begin{aligned} QFT|j\rangle &= QFT|j_1 j_2 \dots j_n\rangle \\ &= \tfrac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_n}|1\rangle) \otimes \tfrac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle) \otimes \dots \otimes \tfrac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle). \end{aligned}$$

Note that the above representation and the equation (*) are equivalent. We leave the proof to the reader as an exercise. This interpretation allows us to construct an efficient quantum circuit for computing QFT. Let us first define the unitary transformation R_k as follows.

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

The efficient circuit for QFT is given as follows.

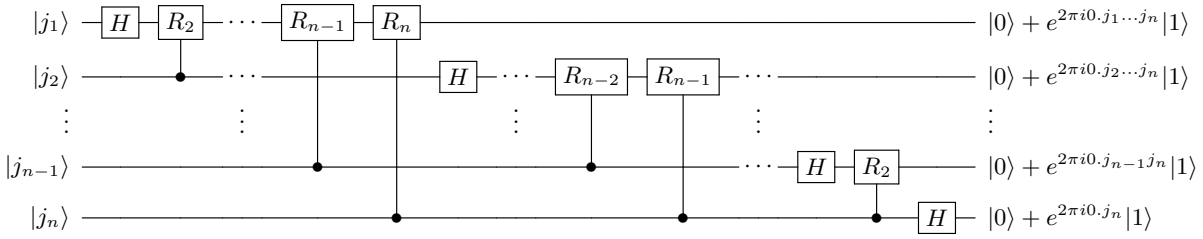


Figure 37: Efficient circuit for QFT.

To see that this circuit computes the quantum Fourier transform, consider what happens when the state $|j_1, \dots j_n\rangle$ is given as an input. Applying the Hadamard transform to the first qubit produces the state

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1}|1\rangle)|j_2 \dots j_n\rangle$$

since $e = 2\pi i 0.j_1 = -1$ when $j_1 = 1$, and $e^{2\pi i 0.j_1} = 1$ otherwise. Applying the controlled-R_2 gate produces the state

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle)|j_2 \dots j_n\rangle.$$

We continue applying the controlled-R_3, R_4 through R_n gates, each of which adds an extra bit to the phase of the coefficient of the first $|1\rangle$. At the end of this procedure, we get the state

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle)|j_2 \dots j_n\rangle.$$

Next we perform a similar procedure on the second qubit. The Hadamard transform puts us in the state

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 \ldots j_n}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2}|1\rangle)|j_3 \ldots j_n\rangle,$$

and the controlled-R_2 through R_{n-1} gates yield the state

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 \ldots j_n}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle|j_3 \ldots j_n\rangle$$

If we continue in this fashion for each qubit, giving a final state

$$\frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle) \otimes \ldots \otimes (|0\rangle + e^{2\pi i 0.j_n}|1\rangle).$$

We then swap (which was omitted in the figure) the qubits to reverse the order of the qubits. After the swap operation, the state becomes

$$\frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i 0.j_n}|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle) \otimes \ldots \otimes (|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle).$$

So how many gates does the circuit use? We start by doing a Hadamard transform and $n-1$ conditional rotations on the first qubit, making a total of n gates. This is followed by a Hadamard transform and $n-2$ conditional rotations on the second qubit, for a total of $n + (n-1)$ gates. This way, we see in the end that we need $n + (n-1) + \ldots + 1 = \frac{n(n+1)}{2}$ many gates.

QFT from another perspective. Now let us look at from another point of view to understand the quantum Fourier transform better. QFT of dimension N, where N is the power of 2, is actually defined as follows.

$$QFT_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)^2} \end{bmatrix}$$

Then the matrix for the quantum Fourier transform for an N-dimensional ($\log N$ qubit) quantum system has the N-th roots of unity as its entries. Recall that $\omega = e^{\frac{2\pi i}{N}}$ is the primitive N-th root of unity. In fact, there is an

easy definition for the entries of the matrix. $QFT_N[j,k] = \omega^{jk \mod N}$. To understand what the matrix looks like, consider QFT_8 as an example.

$$QFT_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix}$$

Quantum Fourier transform is very related to the Hadamard transform. In fact, quantum Fourier transform is a generalization of the Hadamard transform. We leave the reader to write QFT_2 and compare it with the Hadamard transform. It will be clear to see that the entries of the Hadamard transform concerns the 2nd roots of unity.

Let us give another example by defining QFT_4. What are the 4th roots of unity? In this case,

$\omega = e^{\frac{2\pi i}{4}} = e^{\frac{\pi i}{2}} = \cos\left(\frac{\pi}{2}\right) + i\sin\left(\frac{\pi}{2}\right) = i$
$\omega^2 = i^2 = -1$
$\omega^3 = i^3 = -i$
$\omega^4 = 1$

Then,

$$QFT_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

We might wonder how this matrix acts on a quantum state. The matrix takes a general superposition state and outputs another superposition state.

$$\frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} = \begin{bmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \beta_3\end{bmatrix}.$$

Let us apply QTF_4 on the two qubit state $|10\rangle$.

$$\frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{bmatrix}\begin{bmatrix}0\\ 0\\ 1\\ 0\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\\ -\frac{1}{2}\\ \frac{1}{2}\\ -\frac{1}{2}\end{bmatrix} = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle.$$

The classical discrete Fourier transform (DFT) has complexity of $O(N^2)$ for the fact that we need to make N many multiplications and additions for N many entries of the given input vector. The Fast Fourier Transform (FFT) algorithm gives the same output by performing $O(N \log N)$ steps which is nearly a quadratic improvement. When it comes to QFT, we have an even better improvement. Suppose for simplicity that $N = 2^n$ for some natural number n. The input to QFT is a superposition of n-qubit states and it outputs another superposition of n-qubit states. It turns out that the complexity of the circuit for QFT can be made as small as $O(n^2)$. Considering that $n = \log N$, the complexity of QFT is $O(\log^2 N)$. This is an exponential improvement over the classical FFT algorithm. There is a catch though. Despite that the output of QFT is a superposition of states, we will not be able to access the superposition unfortunately, as quantum mechanics forbids us from measuring this enormous data. When we measure the qubits, we will just get to see an index i with some probability for the basis state the superposition is projected onto. In other words, we just see a *sample* of the output. Moreover, we have confined our attention to the case $N = 2^n$ for now, but the algorithm can be implemented for arbitrary values of N. The steps of the quantum fourier transform can be summarized as follows:

1. Input: QFT takes as input a superposition of $m = \log M$ qubits $|\psi\rangle = \sum_{j=0}^{M-1} \alpha_j |j\rangle$.

2. Method: Using $O(m^2) = O(\log^2 M)$ quantum operations, perform the quantum Fourier transform to obtain the superposition
$|\psi'\rangle = \sum_{j=0}^{M-1} \beta_j |j\rangle$.

3. Output: A random m-bit number j, where $0 \leq j \leq M-1$, with probability $|\beta_j|^2$.

We will discuss how QFT is implemented, but first let us give some useful properties of QFT.

Properties of QFT. We shall give two important properties of QFT. However, before explaning these properties we should first show that QFT_M is a unitary operator. Note that there are exactly M many M-th roots of unity. Recall that $(\omega^j)^* = \omega^{-j}$. Also note that QFT_M is a symmetric matrix, so the adjoin of QFT_M is simply

$$QFT_M^\dagger[j,k] = \frac{1}{\sqrt{M}}(\omega^{jk})^* = \frac{1}{\sqrt{M}}\omega^{-jk}.$$

Proposition 30. *QFT_M is a unitary operation.*

Proof. To show that QFT_M is unitary, let us multiply QFT_M with $QFT_M^\dagger$.

$$(QFT_M \times QFT_M^\dagger)[j,k] = \frac{1}{M}\sum_{i=0}^{M-1} \omega^{ji}\omega^{-ik} = \frac{1}{M}\sum_{i=0}^{M-1} \omega^{i(j-k)}.$$

If $j = k$, i.e. when we take the diagonal entries, this becomes

$$\frac{1}{M}\sum_{i=0}^{M-1} \omega^0 = \frac{1}{M}\sum_{i=0}^{M-1} 1 = 1.$$

We know from geometric series that

$$\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}.$$

If $j \neq k$, i.e. when we are off diagonal entries, we have a geometric series and the summation is equal to 0. This tells us that the resultant matrix has 1 on the diagonal entries and has 0 elsewhere. So $QFT_M \times QFT_M^\dagger = I$. Therefore, QFT_M is a unitary matrix. □

Now we can look at the other properties of the quantum Fourier transform. The first property is the *convolution multiplication* property (a.k.a. linear shift) . Suppose that we have an N-dimensional quantum state

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$$

such that $\sum |\alpha_i|^2 = 1$. Suppose that we want to apply QFT to this state. The outcome will be another state

$$|\psi'\rangle = \sum_{i=0}^{N-1} \beta_i |i\rangle$$

Now suppose that we shift the input vector one row down as

$$\begin{bmatrix} \alpha_{N-1} \\ \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-2} \end{bmatrix}$$

Now if we apply QFT_N on this new vector, we have

$$\frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} \alpha_{N-1} \\ \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-2} \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \omega \cdot \beta_1 \\ \omega^2 \cdot \beta_2 \\ \vdots \\ \omega^{(N-1)^2} \cdot \beta_{N-1} \end{bmatrix}.$$

The coefficients of the output vector are now multiples of the N-th roots of unity. But the complex roots are of unit length, so they do not change the probability of the outcome. The outcome will be the same as earlier. Shifting the input vector does not change the output probability distribution.

Second property of QFT is the way it treats periodic functions. Recall that a function f is *periodic* if there exists some r such that $f(x+r) = f(x)$. General behavior of QFT, given a periodic superposition of states, is that

it transforms a given periodic superposition state of length M with period r into a superposition of the same length with period $\frac{M}{r}$.

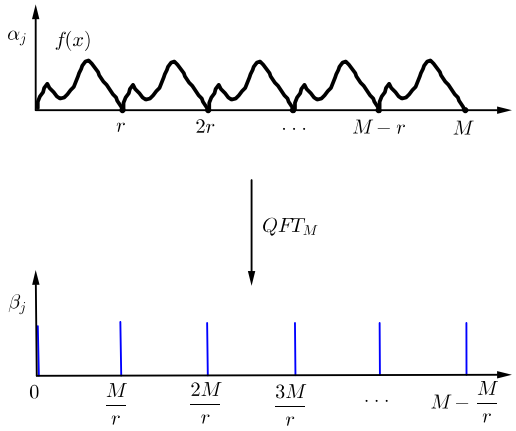
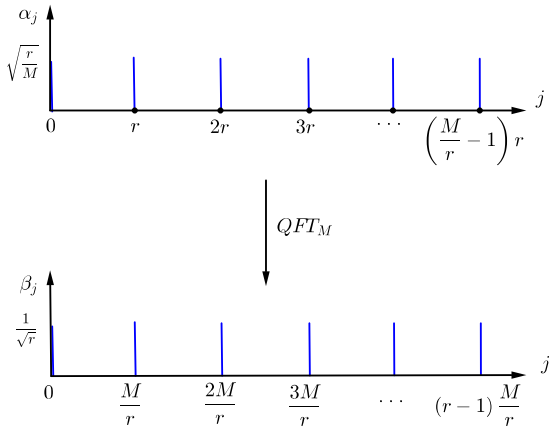


Figure 38: QFT_M maps a periodic superposition with period r to a periodic superposition of period M/r.

GIVE AN EXAMPLE TO A PERIODIC SUPERPOSITION STATE.

But let us suppose a special case. For simplicity we assume that the period r divides M, where M is the dimension of the input vector which is of size at least N^2 such that N is the number we want to factor. The reason why we take M to be of size at least N^2 will be explained later when we describe the algorithm. Suppose that the input to the QFT, given as $|\psi\rangle = (\alpha_0, \alpha_1, \ldots, \alpha_{M-1})$, is such that $\alpha_i = \alpha_j$ whenever $i \equiv j \mod r$, where r is a particular integer that divides M. That is, the vector $|\psi\rangle$ consists of M/r repetitions of some sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{r-1})$ of length r. Moreover, suppose that exactly one of the r numbers $\alpha_0, \alpha_1, \ldots, \alpha_{r1}$ is non-zero, say α_j. Then we say that $|\psi\rangle$ is *periodic with period r and offset j.*



If the input vector is an M-dimensional periodic superposition with period r, then using QFT_M we can compute its period by the following property:

QFT takes as input a superposition with period r with some offset, for some r that divides M. Then it outputs a superposition with period M/r without an offset. The outcome of the measurement will be any of the r multiples of M/r.

Let $|\psi\rangle = (\alpha_0, \alpha_1, \ldots, \alpha_{M-1})$ be a periodic vector with period r and with no offset, i.e. the only non-zero terms are $\alpha_0, \alpha_r, \alpha_{2r}, \ldots$. That is, let

$$|\psi\rangle = \sqrt{\frac{r}{M}} \sum_{j=0}^{M/r-1} |jr\rangle.$$

Then, the quantum Fourier transform of $|\psi\rangle$ is a periodic vector $|\psi'\rangle$ with period $\frac{M}{r}$ and with no offset. That is,

$$QFT_M|\psi\rangle = |\psi'\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \left|\frac{jM}{r}\right\rangle$$

The action of an M-dimensional QFT on a given superposition can be described as in Figure 39.

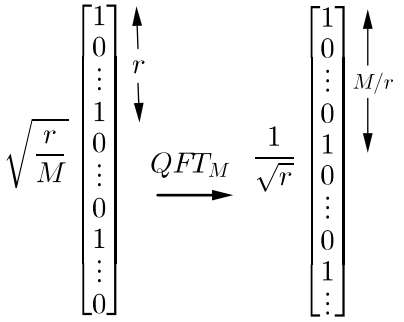


Figure 39: Action of QFT_M on the amplitudes. Given a vector with only non-zero entries on the multiples of r, QFT_M outputs a vector (superposition of states) whose only non-zero entries are the multiples of M/r.

The Algorithm.

The algorithm uses two registers. Suppose that N is the integer we want to factorize. The first register stores the binary representation of an integer M such that $M = N^2$. So it stores m-qubits where $m = \log_2 M$. The second register has at least $\log_2 N$ qubits for storing a number $\mod N$.

Suppose that N is the number we want to factor. We take M to be as large as N^2. The reason is the following. Here, we compute $M > N^2$ values of the modular function $f(x)$ in parallel. Since $r < N$ and there can only be r different function values, the period r must manifest itself in the resulting sequence of N^2 function values stored in the second register.

Shor's algorithm, whose circuit model is shown in Figure 40, can be given as follows.

1. We start with initializing both registers to $|0\rangle$.

2. Apply quantum Fourier transform on the first register and obtain

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle|0\rangle$$

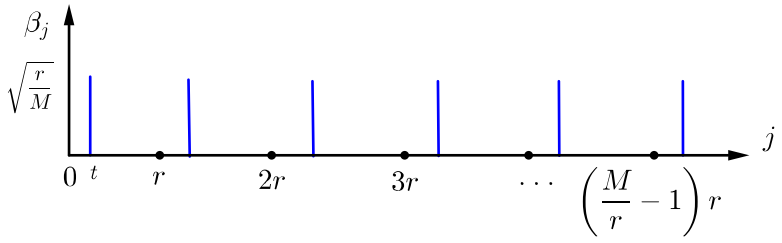
3. Next we apply a quantum gate $U_{f_a(x)}$ that implements the modular exponentiation function $f_a(x) = a^x \mod N$, where a is randomly chosen

integer such that $gcd(a, N) = 1$. We get the state

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |f(x)\rangle$$

-QUANTUM CIRCUIT FOR $U_{f_a(x)}$ WILL COME HERE-

4. We then measure the second register and obtain a random integer a^x mod N. As soon as we measure it, since we are performing a partial measurement, everything in the superposition in the first register which is inconsistent with the measurement outcome will disappear. The only states left in the superposition, in the first register, are the integers x whose a^x mod N is same as the measured value. Now the only non-zero values are multiples of r plus some offset t such that t is a random integer between 0 and $r - 1$. For simplicity, we assume r divides M. We shall investigate the general case later.



The state of the first register after measuring the second register is

$$\frac{1}{\sqrt{\frac{M}{r}}} \sum_{j=0}^{\frac{M}{r}-1} |jr + t\rangle$$

Notice that the distance between the non-zero values is exactly the period r, the value we are looking for. Can we get anywhere by measuring the first register? It's no good, because all we will get is a random point, with no correlation across independent trials (because the offset t is random).

Here's what Shors algorithm does next. This will be the critical step in the algorithm.

5. Apply QFT_M to the first register. The result of this step may not be immediately clear. So let us discuss what happens here.

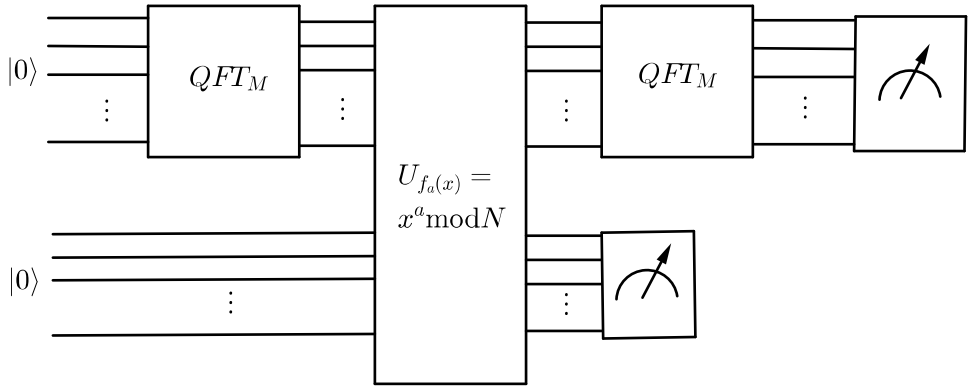


Figure 40: Circuit model of Shor's algorithm.

We may consider the action of QFT_M as

$$\frac{1}{\sqrt{\frac{M}{r}}} \sum_{j=0}^{\frac{M}{r}-1} |jr+t\rangle \xrightarrow{QFT_M} \sum_{j=0}^{M-1} \beta_j |j\rangle$$

We have to figure out what β_j looks like for every j. Let us examine the case when j is a multiple of $\frac{M}{r}$. That is, we look at $\beta_{\frac{kM}{r}}$. It will get a contribution from every part of the input vector, the normalization factor of QFT_M and a phase of the products of the $|jr\rangle$ and $\left|\frac{kM}{r}\right\rangle$.

$$
\begin{aligned}
\beta_{\frac{kM}{r}} &= \sum_{j=0}^{\frac{M}{r}-1} \frac{1}{\sqrt{\frac{M}{r}}} \cdot \frac{1}{\sqrt{M}} \cdot \omega^{jr \cdot \frac{kM}{r}} \\
&= \sum_{j=0}^{\frac{M}{r}-1} \sqrt{\frac{r}{M}} \cdot \frac{1}{\sqrt{M}} \cdot \omega^{jkM}. \text{ Since any multiple of } \omega^M \text{ is 1, we have} \\
&= \sum_{j=0}^{\frac{M}{r}-1} \sqrt{\frac{r}{M}} \cdot \frac{1}{\sqrt{M}} \cdot 1 \\
&= \frac{M}{r} \cdot \left(\sqrt{\frac{r}{M}} \cdot \frac{1}{\sqrt{M}} \right) \\
&= \frac{M}{r} \cdot \frac{\sqrt{r}}{M} = \frac{\sqrt{r}}{r} = \frac{1}{\sqrt{r}}
\end{aligned}
$$

Observe that $\beta_j = 0$ for $j \neq \frac{kM}{r}$. So we have a constructive interference at phase values β_j for $j = \frac{kM}{r}$, and destructive interference at phase values β_j for $j \neq \frac{kM}{r}$.

EXPLAIN THE PHASE SHIFTING A BIT MORE HERE...

6. Now that we have a superposition with period $\frac{kM}{r}$, i.e. multiples of $\frac{M}{r}$, we measure the first register and obtain a random multiple of $\frac{M}{r}$, say $\frac{kM}{r}$, where k is a random number from 0 to $r-1$. It is easy to see that $gcd(k, \frac{M}{r}) = 1$. If so then computing $gcd(\frac{kM}{r}, M)$ will give us $\frac{M}{r}$. Since we know M, from $\frac{M}{r}$, we get r by dividing M to $\frac{M}{r}$. Repeating the calculation $O(\log r) < O(\log N)$ times, one can amplify the success probability of finding r to get as close to one as desired.

To be more precise, we state the following lemma.

Lemma 31. Suppose that s independent samples are drawn uniformly from $0, \frac{M}{r}, \frac{2M}{r}, \ldots, \frac{(r-1)M}{r}$. Then with probability at least $\frac{1-r}{2s}$, the greatest common divisor of these samples is $\frac{M}{r}$.

Proof. The only way this can fail is if all samples are multiples of $\frac{jM}{r}$, where j is some integer greater than 1. We fix any integer $j \geq 2$. The probability that a specific sample is a multiple of $\frac{jM}{r}$ is at most $\frac{1}{r} \leq \frac{1}{2}$, and so the probability that all samples are of multiples of $\frac{jM}{r}$ is $\frac{1}{2^s}$.

So far we have been thinking about a particular number j; the probability that this bad event will happen for some $j \leq k$ is at most equal to the sum of these probabilities over the different values of j, which is no more than $\frac{k}{2^s}$. □

We can make the failure probability as small as we like by taking s to be an appropriate multiple of $\log M$.

Complexity analysis. The best known classical algorithm for factorizing the prime numbers of a given integer is known as *Field Sieve algorithm* that works in time $2^{O(\sqrt[3]{\log n})}$. Let $n = \log N$ be the number of bits of the input integer N. The running time of the algorithm is dominated by the number of repetitions for steps given above, and the number of repetitions, as we noted, is $O(\log N) = n$. Since modular exponentiation takes $O(n^3)$ steps (see Dasgupta, Papadimitriou, and Vazirani, 2006 [**?**], Section 1.2.2), and the quantum Fourier transform takes $O(n^2)$ steps, the total running time of the algorithm is $O(n^3 \log n)$.

The General Case.

We earlier made the assumption that r divides M. However, it may be very well be the case that the period does not divide M at all. We need to show that in the general case, the algorithm still works. If r does not divide M, we may ask what is the probability of observing some p which is *close* to a multiple of $\frac{M}{r}$. We claim that

$$\frac{p}{M} \sim \frac{d}{r}.$$

For if

$$M \left| \frac{p}{M} - \frac{d}{r} \right| = \left| p - \frac{dM}{r} \right|$$

is small enough and $gcd(d, r) = 1$, then $\frac{d}{r}$ is a convergent of $\frac{p}{M}$, and all the convergents of $\frac{p}{M}$ can be found efficiently using Euclid's *continued fractions algorithm*. That is, we can show that $\frac{d}{r}$ is the best approximation to $\frac{p}{M}$ with a denominator as small as r (smaller than $\frac{1}{2r^2}$). We then can construct $\frac{d}{r}$ using the mentioned method.

A *continued fraction* is an expression of the form

$$[a_0, a_1, a_2, a_3, \ldots] \equiv a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ddots}}}$$

where a_i are integers. If the expression contains a finite number of terms, it is called a *finite continued fraction*. If the expression contains an infinite number of terms, it is called an *infinite continued fraction*. For our purpose we will be working with finite continued fractions. We define the *n-th convergent* to this continued fraction to be $[a_0, a_1, \ldots, a_n]$.

Let us give an example. We apply Euclid's algorithm to decompose the fraction $\frac{263}{189}$ into continued fraction as follows.

$263 = 1 \cdot 189 + 74,$
$189 = 2 \cdot 74 + 41,$
$74 = 1 \cdot 41 + 33,$
$41 = 1 \cdot 33 + 8,$
$33 = 4 \cdot 8 + 1,$

and the algorithm terminates at the next iteration giving $gcd(263, 189) = 1$. By divisions we obtain the continued fraction

$$\frac{263}{189} = 1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{1}{8}}}}}$$

Phase estimation. We shall now look at phase estimation. Let us first recall the action of Hadamard transform used in the previous algorithms. The Hadamard transform is used to get an information encoded in the relative phases of a state. The Hadamard gate is self-inverse so it does the opposite as well, i.e. encoding information into the phases. Let us consider the Hadamard transform H on the basis state of one qubit. Recall that

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

$$= \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}} (-1)^{xy} |y\rangle.$$

We may think of the Hadamard transform as having encoded information about the value of x into the phases between the basis states $|0\rangle$ and $|1\rangle$. Since the transform is self-inverse we have,

$$H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^x}{\sqrt{2}}|1\rangle\right) = |x\rangle.$$

Here, the Hadamard transform can be thought of as decoding the information about the value of x that was encoded in the phases. The n-qubit Hadamard gate acting on n-qubit basis state $|\mathbf{x}\rangle$ gives the similar result as we saw earlier. That is,

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

Applying the same gate will give us back the original input state $|\mathbf{x}\rangle$. Unfortunately, $(-1)^{\mathbf{x}\cdot\mathbf{y}}$ are phases of a very particular form. In general, a phase is a complex number of the form $e^{2\pi i\omega}$, for any real number $\omega \in (0,1)$. The phase (-1) actually corresponds to when $\omega = \frac{1}{2}$. The n-qubit Hadamard transformation is not able to fully access information that is encoded in more general ways. We should generalize this transformation to determine the information encoded in phases in the general form. Suppose that we are given a state

$$\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i\omega y} |y\rangle,$$

where $\omega \in (0,1)$, and for simplicity suppose $N = 2^n$ for some natural number n.

5 Quantum Error Correction

The difference between a physical computing device and a mathematical model of computation is that abstract mathematical models are immune to physical distortions caused by the surrounding factors due to that mathematical models of computation are merely idealizations of a physical device in a perfect idealized ground. It is however, inevitable that a physical computing device will have an amount of error caused by electromagnetic noises or other factors coming from the outside. If individual steps in a computation succeed with probability p, then a computation involving t sequential steps will have a success probability that decreases exponentially as p^t.

Many classical computing devices use error correcting codes to perform detection of and recovery from errors. Quantum computers are more suspectible to errors than classical digital computers since quantum systems are more delicate and more difficult to control. If it is possible to build large-scale quantum computers, having a theory of quantum error correction will be necessary to have. This way it would be actually possible to do computation with large-scale quantum computers even with the presence of errors. The most straightforward way of protecting a classical bit is to apply the *repeating state* solution. If b is a bit, we may repeat b three times bbb and obtain a codeword. Whenever we want to decode the codeword and obtain the encoded bit, we can take the majority value of the three bits. Unfortunately, we cannot do such straightforward implements in quantum computing for several reasons.

(i) Repeating any state is not possible in quantum computing due to the No-Cloning Theorem.

(ii) Classical errors only consist of bit-flip errors, whereas quantum errors are continuous.

(iii) Measurements that test whether the state is correct can collapse the state and cause us to lose information.

It is however still possible to overcome these issues once we model the errors.

5.1 Classical case

The theory of classical error correction contains three concepts.

(i) the characterization of the error model.

(ii) the introduction of redundancy through encoding.

(iii) Error recovery procedure.

The Error Model.

Understanding the *error model* is the first step in protecting our information. An error model describes the evolution of set of bits. We call *channel* to make an analogy where the errors occur. Ideally, we would like the state of our bits to be unaffected by the channel. That is, we do not want our bits to be changed when they are stored or being moved from one place to another. We say that an *error-free* channel is an *identity channel*, in the sense that the bits are not affected by anything. The channel provides the description of the error when such error occurs on bits being stored or moved around. We ultimately want to consider errors that occur during a computation. It is a good idea to begin with a very simple error model, called the *bit-flip* error when transmitting bits through a channel. In the bit-flip model, the state of a bit is *flipped* with probability p, and is unaffected with probability $1 - p$.

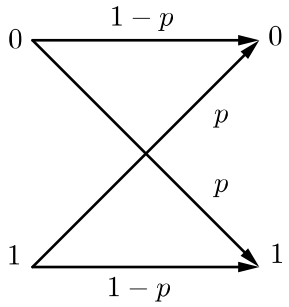


Figure 41: Classical bit-flip channel.

When errors described by a given model act on a register of bits in a circuit, we will show this by a block label $\mathcal{E}^C$. The superscript C denotes that the error model is classical rather than quantum. It might also be the case that $\mathcal{E}^C$ is composed of different operations $\mathcal{E}_i^C$, each $\mathcal{E}_i^C$ corresponding to a different error model occuring with probability p_i.

Encoding.

If we have a description of the error model, we can encode information so that the bits that are to be processed are robuts against these errors.

This can be done by adding some extra bits to the *logical bit* that we wish to protect, and as a result, transforming the resultant string into an *encoded bit*. The string of bits corresponding to an encoded bit is called a *codeword*. The set of codewords (one for each of the two possible bit values 0 and 1) is called a *code*.

The codewords are designed to add some redundancy to the logical bits they represent. But the idea is that even if the errors corrupt some of the bits in a codeword, the remaining part will contain enough information so that the logical bit can be recovered. Of course it should be noted that we encode the logical bit prior to any occurence of the error.

In fact, we may generalize this to encode logical strings of n bits of length. A logical string b of n bits of length can be encoded by adding m ancillary bits (fixed in a known state, say $0 \ldots 0$ for convenience). Then the transformed string becomes an $(n+m)$-bit codeword which we shall denote it by b_{enc}. The process of mapping logical strings b to their corresponding codewords b_{enc} is called the *encoding* opertion which will be denoted by G_{enc}.

Error Recovery.

The codeword b_{enc} will be subjected to some errors by the error model, hence outputting some string b^*_{enc}. Error *recovery*, denoted by R^C, is the operation of recovery of the logical string b from b^*_{enc}. The entire process is shown in the figure below.

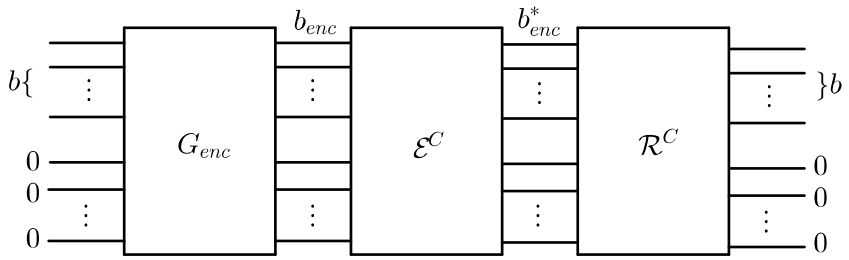


Figure 42: Classical Error Recovery.

The recovery operation must be able to unambiguously distinguish between codewords after errors have acted on them. If we assume that specific errors are represented by $\mathcal{E}^C_i$, a recovery operation should work correctly from some subset of these errors, which we call the *collectable errors*. Given

a code, for a set of errors to be correctable by that code, we must have

$$\mathcal{E}_i^C(k_{enc}) \neq \mathcal{E}_j^C(l_{enc}), \quad \forall k \neq l \qquad (*)$$

where k and l are logical strings encoded into the codewords k_{enc} and l_{enc} and i, j range over the correctable errors. The equation (*) is the condition for classical error correction which says that when any error acts on two different codewords, the resulting strings are never equal.

5.2 Quantum case

We will summarize some of the essential parts of quantum error correction chapter in Nielsen and Chuang's book. We start with a very simple quantum error correction. Suppose we have two parties, Alice and Bob. Assume Alice wants to send quantum information via some noisy communication channel. We consider the bit-flip error and suppose that the noise acts on each qubit independently. Given a qubit, the noise flips the qubit with probability p and leaves it unchanged with probability $1 - p$.

Suppose Alice wishes to send a single qubit state $|\phi\rangle = a|0\rangle + b|1\rangle$ to Bob through the bitflip noise channel. Alice prepares two further qubits each in state $|0\rangle$. She then encodes her single qubit into a compound state of three qubits, by two CNOT gates. These three qubits are sent to Bob. At the receiving end, Bob decodes the compound three qubit state by extracting the error information and correcting on the basis of this information. The correction is a bitflip (NOT gate) operation applied to one or none of the qubits and then leaving Bob with a single qubit in state $|\phi\rangle$ with probability $1 - O(p^2)$. The figure below summarizes the method.

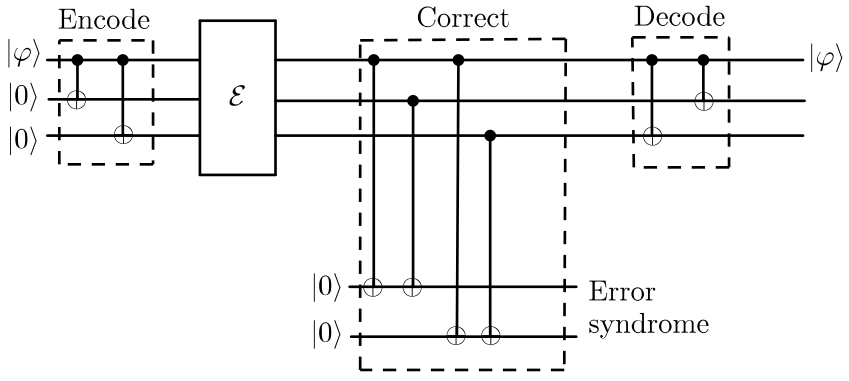


Figure 43: Three qubit code.

Suppose Alice wants to send $a|0\rangle + b|1\rangle$. With the additional qubits, the joint state becomes

$$a|000\rangle + b|100\rangle.$$

Alice now encodes the state by applying CNOT gate and produces

$$a|000\rangle + b|111\rangle.$$

Bob receives the three state qubit through a noisy channel. The state would be one of the following.

State	Probability	Change
$a\|000\rangle + b\|111\rangle$	$(1-p)^3$	no change
$a\|100\rangle + b\|011\rangle$	$p(1-p)^2$	1st bit
$a\|010\rangle + b\|101\rangle$	$p(1-p)^2$	2nd bit
$a\|001\rangle + b\|110\rangle$	$p(1-p)^2$	3rd bit
$a\|110\rangle + b\|001\rangle$	$p^2(1-p)$	1st and 2nd bits
$a\|101\rangle + b\|010\rangle$	$p^2(1-p)$	1st and 3rd bits
$a\|011\rangle + b\|100\rangle$	$p^2(1-p)$	2nd and 3rd bits
$a\|111\rangle + b\|000\rangle$	p^3	all bits

Bob now prepares two more qubits prepared in state $|00\rangle$, referred to as an *ancilla*. Bob uses the ancilla to gather information about the noise. After applying the CNOT gate, the total state of all five qubits is now

State	Probability
$(a\|000\rangle + b\|111\rangle)\|00\rangle$	$(1-p)^3$
$(a\|100\rangle + b\|011\rangle)\|11\rangle$	$p(1-p)^2$
$(a\|010\rangle + b\|101\rangle)\|10\rangle$	$p(1-p)^2$
$(a\|001\rangle + b\|110\rangle)\|01\rangle$	$p(1-p)^2$
$(a\|110\rangle + b\|001\rangle)\|01\rangle$	$p^2(1-p)$
$(a\|101\rangle + b\|010\rangle)\|10\rangle$	$p^2(1-p)$
$(a\|011\rangle + b\|100\rangle)\|11\rangle$	$p^2(1-p)$
$(a\|111\rangle + b\|000\rangle)\|00\rangle$	p^3

Bob measures the two ancilla bits in the canonical basis. This gives him two classical bits of information. We call this information the *error*

syndrome. It will help us to diagnose the errors in the received qubits. Bob's next action is as follows

Measured syndrome	Action
00	do nothing
01	flip the third qubit
10	flip the second qubit
11	flip the first qubit

Suppose for example that Bob's measurements give 10. Examining the table we gave above, the state of the received qubits must be either $a|010\rangle + b|101\rangle$ with probability $p(1-p)^2$ or $a|101\rangle + b|010\rangle$ with probability $p^2(1-p)$. Since the former is more likely, Bob corrects the state by applying the NOT gate to the second qubit. Thus he obtains either $a|000\rangle + b|111\rangle$ or $a|111\rangle + b|000\rangle$. Finally, to extract the qubit which Alice sent, Bob applies CNOT gates on the third and second qubits taking the first qubit as the target qubit, obtaining either $(a|0\rangle + b|1\rangle)|00\rangle$ or $(a|1\rangle + b|0\rangle)|00\rangle$. Therefore, Bob has either the exact qubit sent by Alice, or the negation (NOT) of it. Bob does not know which he has, but the important point is that this method has a probability success greater than $1 - p$. The failure probability is the probability that at least two qubits are corrupted by the channel, which is $3p^2(1-p) + p^3 = 3p^2 - 2p^3$, i.e. less than p (as long as $p < 1/2$).

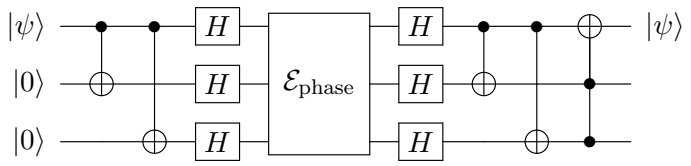
The probability that Bob fails to obtain Alice's original state is $O(p^2)$, whereas it would have been $O(p)$ if no error correction method had been used. With more qubits the same basic idea lead to much more powerful noise supression. The result is already very good. By using just three times as many qubits, we reduce the error probability by a factor $\sim 1/3p$. That is a factor ~ 30 for $p = 0.01$, ~ 300 for $p = 0.001$ and so on.

Another type of error which is more exclusive to quantum computation is the *phase-flip* error. The phase flip error changes the relative phase of a qubit $a|0\rangle + b|1\rangle$ into $a|0\rangle - b|1\rangle$. This kind of error is not found in classical error correction as there is no corresponding phenomena in classical computing for the phase value. However there is an easy way to turn the phase flip channel into a bit flip channel. Suppose we work in the sign basis

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

as logical zero and logical one states, denoted by $|0_L\rangle$ and $|1_L\rangle$ respectively. All operations are performed same as the bit-flip error except that now we are working under the sign basis. To accomplish the basis change we simply apply the Hadamard transform and its inverse (which is also the Hadamard transform) at appropriate points in the procedure. Recall that the Hadamard transform can be used to perform change of basis between the sign basis and the Canonical basis. The quantum circuit of the phase-flip code can be seen in the figure below.



The Shor code. A very nice error correction code for protecting a qubit from an *arbitrary* effect is known as the *Shor code*, named after the inventor himself. This is a 9-qubit code which utilizes the three qubit phase flip and the bit flip code. First we encode the qubit using the phase flip code: $|0\rangle \mapsto |++\rangle$, $|1\rangle \mapsto |--\rangle$. Next we encode each of these qubits using three qubit bitflip code: $|+\rangle$ is encoded as

$$\frac{(|000\rangle + |111\rangle)}{\sqrt{2}}$$

and the state $|-\rangle$ is encoded as

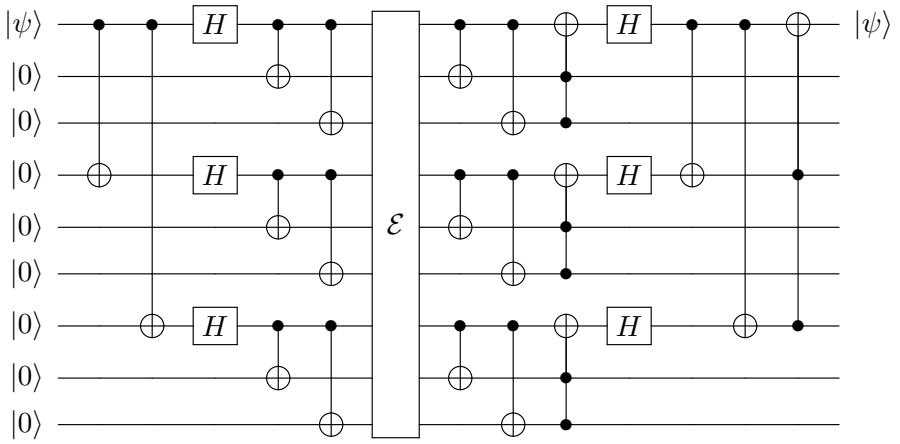
$$\frac{(|000\rangle - |111\rangle)}{\sqrt{2}}.$$

The result is a nine qubit code, with codewords given by

$$|0\rangle \mapsto |0_L\rangle = \frac{|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}$$

$$|1\rangle \mapsto |1_L\rangle = \frac{|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}$$

The quantum circuit of the Shor code is given in figure below.



So how does the Shor code correct the phase flip and bit flip errors on any qubit? Suppose a bit flip occurs on the first qubit. As for the bit flip code, we perform a measurement of Z_1Z_2 (that is $Z \otimes Z \otimes I$) to compare the first two qubits, and find if they are different.[6] If this is the case, we conclude that a bit flip error must have occurred on the first or second qubit. Next we compare the second and third qubit by performing a measurement of Z_2Z_3 (that is, $I \otimes Z \otimes Z$). If we find that they are the same, it could not have been the second qubit which is flipped. We conclude that the first qubit must have been flipped. We recover from the error by flipping the first qubit back to its original state.

We apply a similar approach with the phase flip on the qubits. Suppose a phase flip occurs on the first qubit. Such a phase flip error flips the sign of the first block of qubits, changing $|000\rangle + |111\rangle$ to $|000\rangle - |111\rangle$ and vice versa. Syndrome measurement begins with comparing the sign value of the first and second blocks of three qubits, just as syndrome measurement for the phase flip code began by comparing the sign of the first and second qubits. For example, $(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)$ has the same sign in both blocks, while $(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)$ has different signs. When a phase flip occurs on any of the first three qubits we find that the signs of the first and second blocks are different. Finally we compare the sign of the second and third blocks of qubits. If we find that these are the same, we conclude

[6] Spectral decomposition of Z_1Z_2 tells us that actually comparing the second and third qubit is just applying the operator Z_1Z_2. Since the eigenvalues of this matrix are 1 and -1, measuring it will return one of them. Apparently, it returns -1 if the first bit and the second bit are different; returns 1 if they are the same.

that the phase must have been flipped in the first block of three qubits. We recover from this by flipping the sign in the first block of three qubits back to its original state.

Not only does the Shor code correct both bit flip and phase flip errors, but it also corrects an arbitrary error occurs on a single qubit. The error could be as tiny as rotation about the z-axis of the Bloch sphere, or it could be as disastrous as replacing the entire qubit with garbage. This means that the continuum of errors on a single qubit can be corrected by correcting only a discrete subset of these errors.

6 Models of Computation and Complexity

Mathematical models of computation are abstract devices to study the limits of computability and computing power. As we saw in Chapter 4, although quantum algorithms could be defined using circuit models, in this section we shall introduce another type of uniform approach which uses abstract models of computation. For this we take Turing machines as a standard model of computation and define the quantum counterpart of this model.

First let us give some notions from formal language theory. An *alphabet* is a finite set of symbols. A *string* is a finite sequence of symbols over some alphabet. A *language* is a set of strings. Given two strings w and u, wu denotes the *concatenation* of w and u. The *length* of a string w is denoted by $|w|$. The *empty string* ϵ is the unique string of length 0. Given an alphabet Σ, Σ^k denotes the set of strings of length k over the alphabet Σ. We denote the set of all strings over Σ by Σ^* (this should not be confused with the notation for the complex conjugate).

6.1 Turing machines

We now define an abstract model of computation called *Turing machine* which basically consists of a control unit having finitely many states and an infinite tape on which we write symbols and which moves around the tape cells. The control unit carries the computation by following the given transition rules. The formal definition of a Turing machines is as follows.

Definition 32. A *(deterministic) Turing machine* M is a 6-tuple

$$(Q, \Sigma, \delta, q_0, q_a, q_r)$$

such that Q is a finite set of *states* where $q_0 \in Q$ is the *start state*, $q_a \in Q$ is an *accepting state* and $q_r \in Q$ is a *rejecting state*, Σ is the alphabet, and δ is the *transition function* defined as

$$\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, S, R\}$$

such that L and R respective denote one step *left (L)* , *right (R)*, S denotes *stationary (S)* movement of the tape head.

The reason for calling this machine deterministic is due to the form of the transition function δ. It is required that every pair $Q \times \Sigma$ is mapped to a unique triplet $Q \times \Sigma \times \{L, S, R\}$.

A *configuration* of a Turing machine is a triplet (q, x, y), where $q \in Q$, $x, y \in \Sigma^*$. We interpret a configuration as follows: If $c = (q, x, y)$ is a configuration, we say that the Turing machine is in state q and the tape contains the string xy. If the string y begins with the symbol a, we say that the machine is *reading* the symbol a. To describe how the computation is carried out, we may write $x = w_1 a$ and $y = a_1 w_2$ such that $a, a_1 \in \Sigma$ and $w_1, w_2 \in \Sigma^*$. Therefore, c can be written as $c = (q_1, w_1 a, a_1 w_2)$, and the transition function δ defines a transition rule from one configuration to another such that if $\delta(q_1, a_1) = (q_2, a_2, d)$ then a configuration $c = (q_1, w_1 a, a_1 w_2)$ is transformed to, respectively for the left, stationary and right movement of the tape,

$$c' = (q_2, w_1, a a_2 w_2), \quad c' = (q_2, w_1 a, a_2 w_2), \quad \text{or} \quad c' = (q_2, w_1 a a_2, w_2).$$

depending on which direction of the tape head moves $d \in \{L, R, S\}$.

If δ defines a transition from c to c', then we say that c *yields* c' and we denote this by $c \vdash c'$. Every such yield defines a *computational step*. The two exceptional cases $c = (q_1, w, \epsilon)$ and $c = (q_1, \epsilon, w)$ can be handled by introducing the *blank symbol* # and so extending the definition of δ and replacing (q_1, w, ϵ) with $(q_1, w, \#)$, and replacing (q_1, ϵ, w) with $(q_1, \#, w)$.

A *computation* of a Turing machine with an input $w \in \Sigma^*$ is defined as a sequence of configurations $c_0, c_1, \ldots$ such that $c_0 = (q_0, \epsilon, w)$ and $c_i \vdash c_{i+1}$ for each i. We say that the computation *halts* if the state symbol of some configuration c_i is either q_a or q_r. In the former case, we say that the computation is *accepting*. Otherwise, we say that the computation is *rejecting*.

If a computation of a Turing machine M starting with configuration (q_0, ϵ, w) ends up with a halting configuration (q, w_1, w_2) in t computational steps, we say that T computes $w_1 w_2$ from the input w in *time* t.

Example. Let us define a Turing machine that decides if a given binary string w contains a 1. It is a convension for a Turing machine to start reading from the leftmost symbol of the input string. But let us suppose without loss of generality that we start from the mid section of the input string. We want a Turing machine that accepts the given input w if and only if w contains at least one 1. We define the set of states as $Q = \{q_{start}, q_L, q_R, q_{accept}, q_{reject}\}$. The transition function δ is defined as follows:

δ	0	1	#
q_{start}	q_L, L	q_{accept}	q_{reject}
q_L	q_L, L	q_{accept}	q_R, R
q_R	q_R, R	q_{accept}	q_{reject}

The columns show which symbol we read and the rows denote which state the machine is in. In this setting, the tape head will keep moving to the left until it sees the delimeter symbol # (belonging to the alphabet) and then it will move to the rightmost position. If the machine sees a 1 at any stage of the computation, then it will enter into the accepting state. If it fails to find a 1 until it reaches to the rightmost symbol, then the machine will enter into the rejecting state.

Note that being in the same state is not the same as being in the same configuration. In fact, if a Turing machines enters into the same configuration in two distinct steps then we should know that the machine is in a loop.

Since a Turing machine may not necessarily halt on a given input, it can also be seen as an *enumerator* classifying the set of strings accepted by the machine.

Definition 33. A language L is called *recursively enumerable (computably enumerable)* if there is a Turing machine T such that T accepts w if $w \in L$ and but may not necessarily halt if $w \notin L$. We say that L is *recursive (computable)* if there exists a Turing machine T such that T accepts w if and only if $w \in L$.

We denote the class of recursive languages by **R** and the class of recursively enumerable languages by **RE**. Clearly, by definition, $\mathbf{R} \subseteq \mathbf{RE}$. However, it is a known fact that $\mathbf{R} \neq \mathbf{RE}$. One particularly interesting problem which is known to be undecidable is called the *halting problem*. The halting problem is to decide whether a given Turing machine, on a given input, halts or not. The index set which codes the *halting problem* is recursively enumerable but not recursive. That is, the set

$$K = \{i : \text{The } i\text{-th Turing machine halts on input } i\}$$

is an example to a set which is recursively enumerable but not recursive.

It is widely accepted by the scientific community that Turing machines capture the intuitive notion of effective computability. This belief is known as the *Church-Turing Thesis*. This makes it possible for us to use intuitive

algorithmic desrciptions to write a solution for a problem instead of writing the formal Turing machine description.

A *decision problem* is a problem that has a yes/no answer. All decision problems in **R** by definition are algorithmically *decidable* (or solvable). Most problems are *undecidable*. In fact, decidable problems are infinitely small compared to undecidable problems. Recursively enumerable sets are also called *semi-decidable* as the Turing machine halts if there is a solution, but may not halt otherwise. We will not be interested in classifying undecidable problems. We are more interested in the time complexity classification of decidable problems.

The set of problems that can be computed by deterministic Turing machines in polynomial time, i.e., in n^k steps for some $k \in \mathbb{N}$ with respect to the given input of length n, is denoted by the class **P**.

A *non-deterministic Turing machine* is just like the deterministic counterpart except that the transition function in this case is allowed to map more than one triplets. That is, for a non-deterministic Turing machine, we have that $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{L, S, R\}$ which tells whether it is *possible* for a configuration c to yield a configuration c'. So $(q_1, a_1, q_2, a_2, d) \in \delta$ means that if the machine is in state q_1 reading the symbol a_1, it is possible that the machine will go into state q_2, write a_2 and move its tape head to the d direction.

We denote the class of problems that are solvable by non-deterministic Turing machines in polynomial time by **NP**. Every deterministic Turing machine can be simulated by a non-deterministic Turing machine. So it is clear that $\mathbf{P} \subseteq \mathbf{NP}$. However, it is one of the millennial prize problems in theoretical computer science whether $\mathbf{P} \neq \mathbf{NP}$. Defining the class **co-P** to be the set of all problems whose complements are solvable in polynomial time by a deterministic Turing machine, we know that $\mathbf{co\text{-}P} = \mathbf{P}$. Defining the class **co-NP** to consist of languages whose complements are solvable by non-deterministic Turing machines in polynomial time, it is unknown whether or not $\mathbf{co\text{-}NP} = \mathbf{NP}$ for the same reason that the relationship between **P** and **NP** is unknown. For the same reason that $\mathbf{P} \subseteq \mathbf{NP}$, we have that $\mathbf{P} \subseteq \mathbf{co\text{-}NP}$. We define **PSPACE** to be the set of all problems that can be solved by deterministic Turing machines using a polynomial number of spaces (cells) on the tape. Since a non-deterministic Turing machine can change only one cell per time step, machines that use $p(n)$ time step to solve

a problem cannot use more than $p(n)$ spaces of its infinite tape. Hence, we have **NP** $\subseteq$ **PSPACE** and **co-NP** $\subseteq$ **PSPACE**.

6.2 Probabilistic Turing machines

We now look at a new kind of Turing machine model which is not entirely deterministic or equally non-deterministic, but rather probabilistic.

Definition 34. A *probabilistic Turing machine* (PTM) M is a 6-tuple

$$(Q, \Sigma, \delta, q_0, q_a, q_r)$$

such that Q is a finite set of *states* where $q_0 \in Q$ is the *start state*, $q_a \in Q$ is an *accepting state* and $q_r \in Q$ is a *rejecting state*, Σ is the alphabet, and δ is the *transition function* with the requirement that for all $(q_1, a_1) \in Q \times \Sigma$

$$\sum_{(q_2,a_2,d)\in Q\times\Sigma\times\{L,S,R\}} \delta(q_1, a_1, q_2, a_2, d) = 1.$$

We may assume for convenience that the values of $\delta(q_1, a_1, q_2, a_2, d)$ are rational. If a configuration c yields another configuration c' with probability p, we denote it by $c \vdash_p c'$. Let $c_0, c_1, \ldots, c_t$ be a sequence of configurations such that $c_i \vdash_{p_{i+1}} c_{i+1}$ for each i. Then we say that c_t is *computed* from c_0 in t steps with probability $p_1 p_2 \cdots p_t$. If $p_1 p_2 \cdots p_t \neq 0$, we also say that $c_0 \vdash_{p_1} c_1 \vdash_{p_2} \ldots \vdash_{p_t} c_t$ is a *computation* of a probabilistic Turing machine. So unlike in a deterministic computation, a single configuration can now yield several different configurations with possibly different probabilities that add up to 1.

For any stage t, we use the notation

$$p_1[c_1] + p_2[c_2] + \cdots + p_m[c_m],$$

where $p_i \geq 0$ and $p_1 + \cdots + p_m = 1$, to denote that the system is in state c_i with probability p_i at stage t. All computations can be expressed by the terms of a probabilistic system. Suppose that by the end of t computational steps, a probabilistic Turing machines starting from an initial configuration computes configurations $c_1, \ldots, c_m$ with probabilities $p_1, \ldots, p_m$ such that $p_1 + \cdots + p_m = 1$. We then say that the *total configuration* of a probabilistic machine at time t is a probability distribution

$$p_1[c_1] + p_2[c_2] + \cdots + p_m[c_m]$$

over the basis configuration $c_1, \ldots, c_m$. We can define a vector space, we call it the *configuration space*, that has all of the potential basis configurations as the basis vectors. A general configuration then can be considered as a linear combination in the configuration space with probability factors adding to 1.

Example. Improving our example given earlier for the standard Turing machine for deciding whether or not the given string contains a 1, we may also define a probabilistic Turing machine solving the same problem. The only difference now is that the transitions are non-deterministic with a probability weight for each transition. We allow *false negative* and *false positive* results as the machine is probabilistic in nature. This means that even though the string contains a 1, the machine might give a false negative result saying that it does not contain a 1. Similarly for the false positive case.

δ	0	1	#
q_{start}	$\frac{1}{2} : q_L, L;\ \ \frac{1}{2} : q_R, R$	$1 : q_{accept}$	$1 : q_{reject}$
q_L	$1 : q_L, L$	$1 : q_{accept}$	$1 : q_{reject}$
q_R	$1 : q_R, R$	$1 : q_{accept}$	$1 : q_{reject}$

When the computation starts, fifty percent of the time the tape head moves to the left and fifty percent of the time it moves to the right. If the tape head is initially placed on the mid section of the string, the machine will examine $\frac{n}{2}+1$ boxes and hence will give a correct answer more than half the time. In the worst case scenario, the machine will have to go through $\frac{n}{2}$ time steps.

6.3 Quantum Turing machines

We may want to generalize the notion of computability to the domain of quantum computers. Recall that the *Church-Turing Thesis* tells us that any effectively computable function can also be computed by a classical computer. An extended version of this thesis would cover quantum models of computation

Extended Church-Turing Thesis: Any effectively computable function can also be computed by a quantum computer, i.e. a computer which is quantum mechanical.

We consider a straightforward generalization of a probabilistic Turing machine, replacing the probability factors with transition amplitudes. We

define the *transition amplitude function* as

$$\delta : Q \times \Sigma \times Q \times \Sigma \times \{L, S, R\} \to \mathbb{C}$$

such that $\delta(q_1, a_1, q_2, a_2, d)$ gives the *amplitude* that whenever the machine is in state q_1 reading symbol a_1, it will write a_2, enter state q_2, and move the tape head in the direction of $d \in \{L, S, R\}$.[7]

Definition 35. A *quantum Turing machine (QTM)* is a 6-tuple

$$(Q, \Sigma, \delta, q_0, q_a, q_r),$$

where $q_0, q_a, q_r \in Q$ are the initial, accepting, and rejecting states, respectively. The transition amplitude function satisfies that for any $(q_1, a_1) \in Q \times \Sigma$,

$$\sum_{(q_2,a_2,d)\in Q\times\Sigma\times\{L,S,R\}} |\delta(q_1, a_1, q_2, a_2, d)|^2 = 1.$$

As in any probabilistic computation, a general configuration of a quantum Turing machine is a linear combination

$$\alpha_1|c_1\rangle + \ldots + \alpha_n|c_n\rangle$$

of basis configurations. We take the basis configurations as an orthonormal basis of a Hilbert space. So we can see that the general configuration written above, which is a superposition, is merely a unit length vector in the configuration space. The transition amplitude function determines a linear mapping U_δ in the state space which is required to be a unitary operator. It is known that the unitarity of U_δ can be determined by some locality conditions (called *well-formedness conditions*) on the transition amplitude function. We may form a square matrix U_δ that describes the transition amplitudes going from one configuration to another. The matrix U_δ would be defined as

$$U_\delta = \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & \cdots & c_{0,j} & \cdots \\ c_{1,0} & c_{1,1} & c_{1,2} & \cdots & c_{1,j} & \cdots \\ c_{2,0} & c_{2,1} & c_{2,2} & \cdots & c_{2,j} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{j,0} & c_{j,1} & c_{j,2} & \cdots & c_{j,j} & \cdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \ddots \end{bmatrix}$$

[7] Of course when we write $\mathbb{C}$, we really mean the set of complex numbers whose real and imaginary parts are computable, i.e. given any natural number n, we should be able to algorithmically determine its nth digit.

where $c_{i,k}$ denotes the transition amplitude going from the basis configuration i to k. For every row, all but finitely many entries will be 0. We say that a QTM is *well-formed* if U_δ preserves the inner product in the configuration space, i.e. the set of all configurations.

QTMs work differently than PTMs. Rather than carrying out one of the possibilities, QTM performs all the operations and enters a superposition of all the resulting states. Then it collapses into a single state when it is measured. In fact, if we observe the state and the contents of the tape of the quantum Turing machine after each step, then a quantum Turing machine will be the same as a probabilistic Turing machine. The difference is that when we *do not* observe the state and the contents of the tape, the probabilities of performing one operation followed by another sum up as complex numbers. Hence when we do not observe, we may see some interference and superposition effects on the content of the tape.

Example. We continue with our last example in the previous subsection and solve the same problem using a quantum Turing machine. A QTM solving the substring problem can be defined as follows.

δ	0	1	#
q_{start}	$\frac{1}{\sqrt{2}} : q_L, L;\ \frac{1}{\sqrt{2}} : q_R, R$	$1 : q_{accept}$	$1 : q_{reject}$
q_L	$1 : q_L, L$	$1 : q_{accept}$	$1 : q_{reject}$
q_R	$1 : q_R, R$	$1 : q_{accept}$	$1 : q_{reject}$

Note that this quantum Turing machine does not start its computation by moving either to the right or to the left. It rather moves to the right and to the left *simultaneously*. A typical computation of this machine might look as follows.

$$|\#000q_{start}0010\#\rangle \vdash \tfrac{1}{\sqrt{2}}|\#00q_L00010\#\rangle + \tfrac{1}{\sqrt{2}}|\#0000q_R010\#\rangle \vdash$$
$$\tfrac{1}{\sqrt{2}}|\#0q_L000010\#\rangle + \tfrac{1}{\sqrt{2}}|\#00000q_R10\#\rangle \vdash$$
$$\tfrac{1}{\sqrt{2}}|\#q_L0000010\#\rangle + \tfrac{1}{\sqrt{2}}|\#00000q_{accept}10\#\rangle.$$

Although QTMs are legitimate models of quantum computing, they are very impractical to define quantum algorithms. To define quantum algorithms we usually use the *circuit model* which was introduced in the previous chapters. Many of the algorithms that we learned required us to we apply a Hadamard transform $H^{\otimes n}$ to a string of qubits. Let us show how one would do this with a quantum Turing machine. We basically want to read a binary

string and transform each 0 into the $|+\rangle$ state; transform each 1 into the $|-\rangle$ state. Suppose that a finite binary string w is given on the tape and that the head is positioned on the leftmost symbol of the string. Then, the following transition function applies $H^{\otimes_n}$ on w.

δ	0	1	#
q_{start}	$\frac{1}{\sqrt{2}} : 0, q_{start}, L$ $\frac{1}{\sqrt{2}} : 1, q_{start}, L$	$\frac{1}{\sqrt{2}} : 0, q_{start}, L$ $-\frac{1}{\sqrt{2}} : 1, q_{start}, L$	$1 : q_{stop}$

Let us now discuss about the properties of QTMs. The first thing we should notice is that the transition function of QTM determines a unitary, hence reversible time evolution in the configuration space. Reversibility is not unique to QTMs though. Standard Turing machines can be also made reversible. Bennett [**?**] gave a reversible Turing machine model which uses a three-tape Turing machine with an *input tape*, *history tape*, and *output tape*. Reversibility is obtained by simulating the standard machine on the input tape and at the same time writing the history of the computation on the history tape. Therefore, it is established that whatever is computable by a standard Turing machine is also computable by a reversible Turing machine. Since QTMs are reversible, QTMs are at least as powerful as standard Turing machines. What about the other way around? It is natural to ask whether or not standard Turing machines are as powerful as QTMs. The answer is positive for the same reason that standard Turing machines can simulate probabilistic Turing machines. However, it is very likely that QTMs cannot be efficiently simulated by standard Turing machines, not even by probabilistic ones. The reason why we guess so is because of the algorithmic speedup we achieve in Shor's algorithm. Although there is a polynomial time quantum algorithm, no such polynomial time algorithm and not even a classical probabilistic algorithm has been found yet despite great efforts.

Another natural question to ask is whether or not there exists a *universal quantum Turing machine*. A Turing machine is *universal* if it can simulate the computation of any given Turing machine on a given input. Deutsch [**?**] proved the existence of a universal QTM which can simulate any other QTM with arbitrary precision.

6.4 Complexity classes

Complexity classes are just equivalence classes containing problems that all have the same amount of algorithmic complexity. It is a way to measure the

hardness of a problem. The class **P** is the set of problems which can be solved by deterministic Turing machines in polynomial time. We also said that the class **NP** was the class of problems solvable by non-deterministic Turing machines in polynomial time. This is also the class of problems solvable by probabilistic Turing machines with non-zero probability in polynomial time.

There are a few options for classifying the time complexity of problems solvable by probabilistic Turing machines. The class **BPP** *(bounded error probability polynomial time)* denotes the set of problems solvable by probabilistic Turing machines with the possibility of some errors. More precisely, if M is a probabilistic Turing machine that decides $L \in \mathbf{BPP}$ and x is an input, then M accepts x with probability at least $\frac{2}{3}$ if $x \in L$. Otherwise, M rejects x with probability at least $2/3$. The fraction $2/3$ may just be replaced with another value p as long as p is greater than $1/2$.

A smaller set of problems are those that can be solved with a probabilistic Turing machine that permits false positives but does not permit false negatives. That is, we permit the machine to accept the input even if the input string does not belong in the language of the machine, but we do not permit the machine to reject the string when it really should have been accepted. We define **RP** to be the set of problems which can be solved by a probabilistic Turing machine in polynomial time with only the possibility of giving false negatives. In other words, if M is a probabilistic Turing machine that decides $L \in \mathbf{RP}$ and if x is a string, then M accepts x with probability at least $2/3$ if $x \in L$, and M rejects x with probability 1 if $x \notin L$.

The class **ZPP** *(zero error probability polynomial time)* denotes the set of problems solvable by probabilistic Turing machines with zero error. More precisely, if M is a probabilistic Turing machine that decides $L \in \mathbf{ZPP}$ and x is an input, then there is a less than $1/2$ chance that the machine will finish in a 'do not know" state, otherwise if the machine does know that $x \in L$, then M accepts x with probability 1 and it rejects x with probability 1 if $x \notin L$.

It is known that $\mathbf{co\text{-}RP} \cap \mathbf{RP} = \mathbf{ZPP}$.

For every $L \in \mathbf{BPP}$, we can create a machine that traverses all the computational paths and keeps track of the paths ending in q_{accept} and q_{reject}. There is no reason to save the path once it is calculated, so we might as well reuse the space. Such a machine will take a very long time to calculate an answer, but it will use only a polynomial amount of space.

From this, it can be seen that **BPP** $\subseteq$ **PSPACE**.

We have the corresponding complexity classes for quantum Turing machines. The class **BQP** *(bounded error quantum polynomial time)* is the quantum counterpart of **BPP**. The class **EQP** *exact quantum polynomial time* is the quantum counterpart of **P**. Finally we may define the class **ZQP** to be the quantum analogue of **ZPP**.

As long as the probabilities are rational numbers (or at least computable numbers), we can always simulate probabilistic Turing machines with deterministic Turing machines by performing the possible transitions sequentially. The equivalency of the computational power of deterministic and probabilistic Turing machines entails an important fact that probabilistic Turing machines do not exceed the Church-Turing barrier. They compute the same class of functions as deterministic Turing machines. Therefore, it is clear that we have the following relationships.

$$\mathbf{P} \subseteq \mathbf{ZPP}, \quad \text{and} \quad \mathbf{P} \subseteq \mathbf{BPP}.$$

Here are some other relations between the classical, probabilistic and quantum complexity classes. The proof of these inclusions can be found in [**?**] or [**?**].

Theorem 36. (i) **P** $\subseteq$ **EQP** $\subseteq$ **BQP**.

(ii) **BPP** $\subseteq$ **BQP**.

(iii) **BQP** $\subseteq$ **PSPACE**.

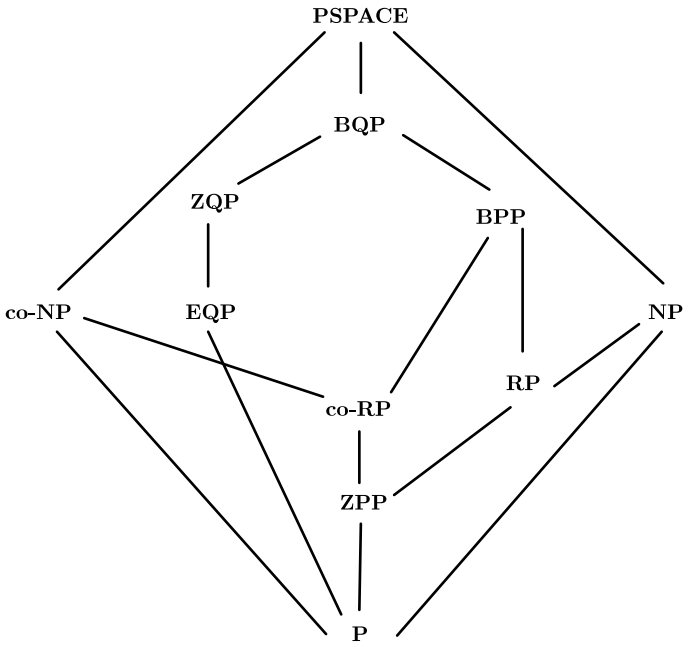


Figure 44: Complexity classes.

A nice overall picture of the complexity classes showing some of the relationships between them can be seen in Figure 44 which is originally given in [**?**].

One natural question to ask is whether or not non-deterministic computations have any advantage over deterministic computations in terms of space complexity. By *Savitch's Theorem*, we know that this is not the case.

Theorem 37 (Savitch's Theorem)**.** For any $f(n) \geq n$, $NPSPACE(f(n)) \subset PSPACE(f(n)^2)$.

Savitch's Theorem, in other words, tells us that if a language L can be decided by a non-deterministic Turing machine with space complexity of the order $f(n)$, then there is also a deterministic Turing machine whose space complexity is at most quadratically worse. This theorem also shows that there is no space complexity analogue of the **P** versus **NP** problem.

The position of the quantum analogue of **NP** is a little more complicated. The class **NQP** is the quantum analogue of **NP** and it was proved that

$$\mathbf{NQP} = \mathbf{CP},$$

where **CP** is the class of decision problems that determine whether the number of accepting computable paths of a non-deterministic Turing machine equals that of rejecting paths. This shows that there is a quantum complexity class can be characterized by a *counting* complexity class. The class **CP** contains for example the graph non-isomorphism problem which is actually not known to be in **NP**. The above equality implies that there is a quantum Turing machine that can accept in polynomial time, with non-zero probability, a description of two graphs if and only if they are non-isomorphic.

6.5 Quantum Finite Automata

This chapter will be a continuation of the previous one regarding models of quantum computation. We look at a simpler model of computation, yet with a richer and more developed theory, called quantum finite automata. It is the only model proven so far that some of its variants outpower the classical counterpart. The reason we began with introducing quantum Turing machines is that they are regarded as the main model to study fundamental questions about the power of quantum computing. However, not much research can be done on quantum Turing machines since they were proven to have equal computational power with that of standard Turing machines and probabilistic Turing machines. So most of contemporary research about models of quantum computation revolves around the quantum counterparts of finite state machines. In this section we assume the reader is familiar with formal languages and how finite state machines work.

6.5.1 Variants of Finite Automata

The simplest model of classical computation is a standard input driven *deterministic finite automaton* (DFA) $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of states, Σ is the input alphabet, q_0 is the initial state, F is the set of final (accepting) states and $\delta : Q \times \Sigma \to Q$ is the transition function such that when the machine is in state q and reads the input symbol a, $\delta(q, a)$ determines the next state of the automaton.

Another way to look at a DFA is a *one-way finite automata* (1-DFA) which has a control unit attached to the input tape by a read-only tape head that can move in a single direction from left to right. A generalization of this model is a *two-way finite automata* (2-DFA) whose tape head can move in both directions. The transition function of a 2-DFA has the form

$$\delta : Q \times \Sigma \to Q \times \{\leftarrow, \downarrow, \rightarrow\}.$$

A *non-deterministic finite automata* (NFA) is a finite automata whose transition function is defined as

$$\delta : Q \times \Sigma \to \mathcal{P}(Q),$$

where $\mathcal{P}(Q)$ denotes the *power set* of Q, i.e. the set of all subsets of Q. We know that DFA and NFA recognize the same class of languages.

Similar to that of Turing machines, as we introduced earlier, two variants of NFA are *probabilistic finite automata* (PFA) and *quantum finite automata* (QFA). The transition function of a PFA has the form $\delta : Q \times \Sigma \times Q \to [0,1]_{\mathbb{Q}}$ such that for each $q \in Q$ and $a \in \Sigma$, we have

$$\sum_{q' \in Q} \delta(q, a, q') = 1.$$

It is known that one-way and two-way deterministic and non-deterministic finite automata accept the same class of languages, namely regular languages. With respect to the bounded error acceptance, 1-PFA and 2-PFA accept regular languages. However, with respect to the unbounded-error acceptance and with arbitrary real probabilities, 1-PFA can also accept non-regular languages.

6.5.2 Quantum Finite Automata

Now we introduce one-way quantum finite automata (1-QFA) as our first primitive model of quantum computation. In fact, two different versions of 1-QFA are known as *measure-many* and *measure-once*. First we give the measure-many version which is just called 1-QFA.

Definition 38. A *one-way quantum finite automata* (1-QFA) A is specified by a finite set of states Q, an alphabet Σ, initial state $q_0 \in Q$, set of accepting states $Q_a \subseteq Q$, set of rejecting states $Q_r \subseteq Q$ such that $Q_a \cap Q_r = \emptyset$ and the transition function

$$\delta : Q \times \Gamma \times Q \to \mathbb{C}_{[0,1]},$$

where $\Gamma = \Sigma \cup \{\#, \$\}$ is the tape alphabet of A, and $\#, \$$ are endmarkers not in Σ. The evolution of A is performed in the inner-product space with the set of basis states $\{|q\rangle : q \in Q\}$ using the linear operators V_a, $a \in \Gamma$, defined by

$$V_a(|q\rangle) = \sum_{q' \in Q} \delta(q, a, q')|q'\rangle,$$

which required to be unitary.

We use in computation of A, the *computational observable* $\mathcal{O}$ which corresponds to the orthogonal decomposition $E_a + E_r + E_n$, where $E_a = \text{span}\{|q\rangle : q \in Q_a\}$, $E_r = \text{span}\{|q\rangle : q \in Q_r\}$, and E_n is the orthogonal complement of $E_a + E_r$. The projection operator that projects a state into the subspace E_a, E_r and E_n is denoted respectively by P_a, P_r and P_n.

A *computation* of a 1-QFA A on a given input $w = \#a_1 a_2 \dots a_n\$$ proceeds as follows. The operator $V_\#$ is first applied to the starting configuration $|q_0\rangle$ and then the observable $\mathcal{O}$ is applied on the resulting configuration $V_\#|q_0\rangle$. This observable projects $V_\#|q_0\rangle$ onto a vector $|\psi'\rangle$ of one of the subspaces E_a, E_r, E_n, with the probability equal to the square of the norm of $|\psi'\rangle$. If $|\psi'\rangle \in E_a$, then we say that the input is *accepted*. If $|\psi'\rangle \in E_r$, then we say the input is *rejected*. If $|\psi'\rangle \in E_n$, then we are in a non-halting state, so after the normalization of $|\psi'\rangle$, the operator V_{a_1} is applied to $|\psi'\rangle$ and after that the observable $\mathcal{O}$ is applied on the resulting vector, and so on. The process continues until we obtain a state in E_a or E_r. When no halting occurs, the computation can be seen as an application of the composed operator

$$V'_{a_n} V'_{a_{n-1}} \dots V'_{a_1} |q_0\rangle,$$

where $V'_{a_i} = P_n V_{a_i}$.

A language L is said to be accepted by a measure-many 1-QFA A with *bounded-error* probability if A accepts any $x \in L$ with probability at least $\lambda + \epsilon$, for some $\epsilon > 0$ and rejects any $x \in L$ with probability at least $\lambda + \epsilon$, where $\lambda \geq 1/2$. L is said to be accepted with *unbounded-error* probability if A accepts any $x \in L$ with probability at least λ and rejects any $x \in L$ with probability at least λ, for $\lambda \geq 1/2$.

This defines the 1-QFA which is *measure-many* by default. This version of 1-QFA is weaker than standard finite automata since there are regular languages that cannot be accepted by measure-many 1-QFA (See Theorem 42).

Definition 39. A *measure-once one-way quantum finite automata* (MO-1QFA) $A = (Q\Sigma, q_0, Q_a, \delta)$ is a 1-QFA without a set of rejecting states such that only one measurement is made at the right endmarker.

We define the bounded and unbounded error acceptance for MO-1QFA the same way as for 1-QFA. It is known that MO-1QFA is strictly weaker than 1-QFA. There is a very nice characterization of the class of languages accepted by MO-1QFA in terms of a well known class of deterministic finite automata.

A deterministic finite automaton $A = (\Sigma, Q, q_0, Q_a, \delta)$ is called a *group finite automaton* (GFA) if for every $q' \in Q$ and $a \in \Sigma$ there exists exactly one $q \in Q$ such that $\delta(q, a) = q'$.

Theorem 40 (Brodsky, Pippinger, 1999)**.** A language can be accepted by an MO-1QFA iff it can be accepted by a GFA.

The same authors also proved an interesting result showing that there exists a non-regular language L for which there exists some MO-1QFA which accepts L with unbounded probability.

An important result proved by Kondracs and Watrous (1997) [**?**] is that 1-QFA do not have greater computational power than standard finite automata.

Theorem 41 (Kondracs and Watrous, 1997)**.** If a language L is accepted by a 1-QFA with bounded-error probability, then L is regular.

In fact, the same authors showed that 1-QFA are weaker than standard finite automata.

Theorem 42 (Kondracs and Watrous, 1997)**.** The regular language $L_0 = \{0, 1\}^*0$ cannot be recognized by a 1-QFA with bounded-error probability.

Two-way quantum finite automata.

We now define the quantum counterpart of two-way finite automata. We will call this 2-QFA and we will note that 2-QFA is provably more powerful than their classical counterparts. The definition of 2-QFA is a bit more complex than of a 1-QFA due to that evolutions have to be unitary. The definition can be given as follows.

Definition 43. A *two-way quantum finite automata* (2-QFA) $\mathcal{A}$ is defined by a finite alphabet Σ, a finite set of states Q, the initial state q_0, the set of accepting states $Q_a \subset Q$ and the set of rejecting states $Q_r \subset Q$ such that $Q_a \cap Q_r = \emptyset$, and the transition function

$$\delta : Q \times \Gamma \times Q \times \{\leftarrow, \downarrow, \rightarrow\} \rightarrow \mathbb{C}_{[0,1]},$$

where $\Gamma = \Sigma = \cup\{\#, \$\}$ is the tape alphabet of $\mathcal{A}$, # and \$ are endmarkers not in Σ. The function δ satisfies the following *well-formedness conditions* (or *unitarity condition*) for any $q_1, q_2 \in Q$ and $a, a_1, a_2 \in \Gamma$ and $d \in \{\leftarrow, \downarrow, \rightarrow\}$:

1. Orthogonality condition.

$$\sum_{q',d} \delta^*(q_1, a, q', d)\delta(q_2, a, q', d) = \begin{cases} 1 & \text{if } q_1 = q_2 \\ 0 & \text{otherwise.} \end{cases}$$

2. Separability condition I.

$$\sum_{q',d} \delta^*(q_1, a, q', \rightarrow)\delta(q_2, a, q', \downarrow) = 0.$$

3. Separability condition II.

$$\sum_{q',d} \delta^*(q_1, a, q', \downarrow)\delta(q_2, a, q', \leftarrow) = 0.$$

4. Separability condition III.

$$\sum_{q',d} \delta^*(q_1, a, q', \rightarrow)\delta(q_2, a, q', \leftarrow) = 0.$$

In short, the separability conditions can be summarized as follows:

$$\sum_{q',d_1,d_2} \delta^*(q_1, a, q', d_1)\delta(q_2, a, q', d_2) = 0,$$

where $d_1, d_2 \in \{\leftarrow, \downarrow, \rightarrow\}$ such that $d_1 \neq d_2$.

It was proved by Kondacs and Watrous [**?**] that the computational power of 2-QFA is at least stronger as that of standard finite automata.

Theorem 44 (Kondracs and Watrous, 1997)**.** Every regular language is accepted by 2-QFA.

The interesting thing about 2-QFA is that there also exists a non-regular language which can be recognized by 2-QFA. The language

$$L = \{0^i 1^i : i \geq 1\}$$

can be recognized by 2-QFA yet cannot be recognized by a standard finite automata. Even some, but not all, non-context-free languages can be recognized by 2-QFA. So the characterization of the language class recognized by 2-QFA looks quite complicated. It is one of the open questions in quantum finite automata theory to give a nice characterization of the class of languages recognized by 2-QFA.